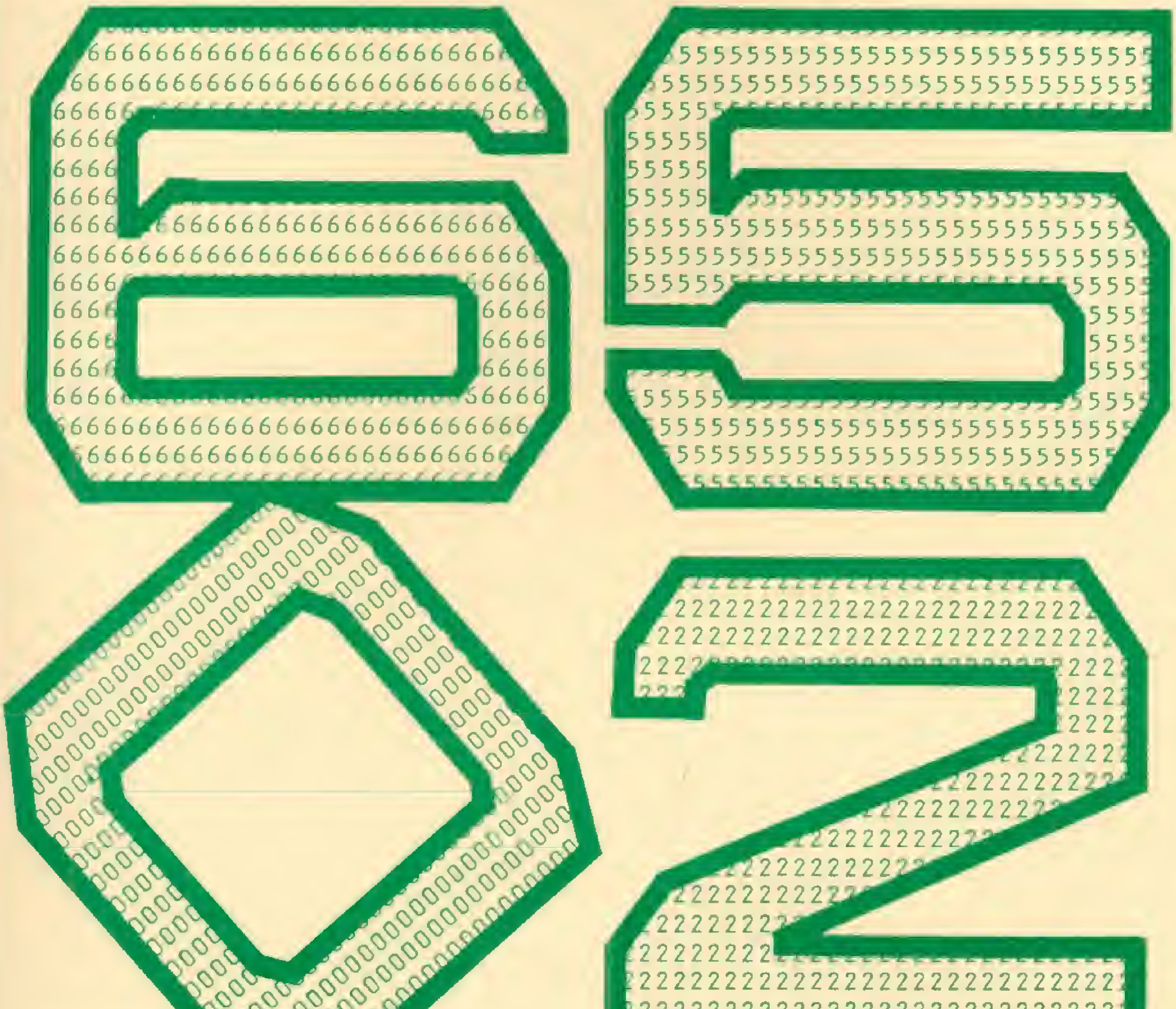
MICRO™

The Magazine of the APPLE, KIM, PET

and Other 6502 Systems



NO 12 May 1979 $1.50

MICRO™

MAY 1979

ISSUE NUMBER TWELVE

TABLE OF CONTENTS

MICRO™

STAFF

Editor/Publisher
Robert M. Tripp

Business Manager
Donna M. Tripp

Administrative Assistant
Maggie Fisher

Circulation Manager
Carol A. Stark

Distribution
Eileen M. Enos
Janet Santaguida

Micro-Systems Lab
James R. Witt, Jr.
Stephen L. Allen

Chief Gofer
Fred Davis

MICRO™ is published monthly by:

The COMPUTERIST®, Inc.
P.O. Box 3
So. Chelmsford, MA 01824

Controlled Circulation postage paid at:

Chelmsford, MA 01824

Publication Number: COTR 395770

Subscription in US: $12.00/12 Issues

ADVERTISER'S INDEX

Please address all correspondence, subscriptions and address changes to:

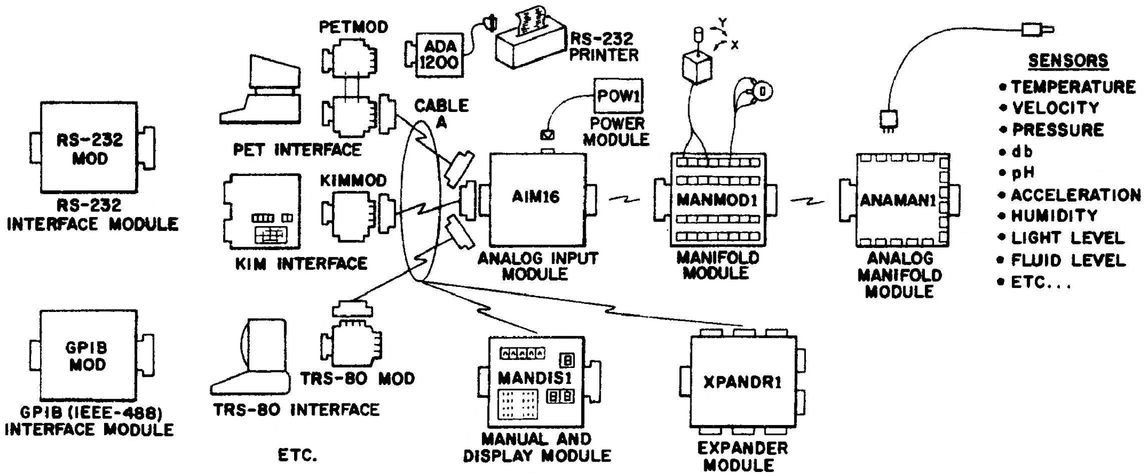
MICRO, P.O. Box 6502, So. Chelmsford, MA 01824

617/256-5515

CONNECTICUT microCOMPUTER, Inc.

150 POCONO ROAD - BROOKFIELD, CONNECTICUT 06804

(203) 775-9659



DAM SYSTEMS by CmC
A complete system of modules to let your computer listen to the real world.

DAM SYSTEMS PRICE LIST

DAM SYSTEMS components

AIM161 - Analog Input Module — $179.00
16 8-bit analog inputs - 100 microsecond conversion time - 3 state output - requires one 8-bit computer output port for control and one 8-bit computer input port for data.

AIM162 - Analog Input Module — $249.00
As above plus: greater accuracy - gold plated contacts - pilot light - switch selectable start, enable and ready polarities.

POW1 - Power Module — $14.95
Supplies power for one AIM16 module.

ICON - Input Connector — $9.95
For connecting analog inputs to the AIM16 - 20 pin card edge connector - solder eyelets.

OCON - Output Connector — $9.95
For connecting the AIM16 to a computer - 20 pin card edge connector - solder eyelets.

MANMOD1 - Manifold Module — $59.95
Use in place of ICON. Screw terminal barrier strips for connecting joysticks, potentiometers, voltage sources, etc. Eliminates the need for soldering. Plugs into the AIM16.

ANAMAN1 - Analog Manifold Module — TBA
Use in place of ICON. Connects DAM SYSTEMS SENSORS to the AIM16 without soldering - sensor cables just plug in. Plugs into the AIM16 or the MANMOD1.

SENSORS — TBA
Sensors for temperature, pressure, flow, humidity, level, pH, motion, etc.

COMPUTER INTERFACES — TBA
For the PET, KIM, TRS-80, etc. Use in place of OCON. Eliminates the need for soldering or special construction.

PETMOD - PET Interface Module — $49.95
Gives two IEEE ports, one user port and one DAM SYSTEMS interface port. Saves wear and tear on the PET's printed circuit board. Also called the PETSAVR.

KIMMOD - KIM Interface Module — $39.95
Gives one application connector port and one DAM SYSTEMS interface port.

CABLE "A" - Interconnect Cables — TBA
Connects computer interface to AIM16, MANDIS1, XPANDR1, etc.

CABLE A24 - Interconnect Cable — $19.95
24 inch cable with interface connector on one end and an OCON equivalent on the other.

MANDIS1 - Manual and Display Module — TBA
Connects between the AIM16 and the computer interface. Allows manual or computer control of the AIM16. Displays channel number and data.

GPIB MOD - GPIB (IEEE-488) Interface — TBA
Allows the DAM SYSTEMS MODULES to be used with the GPIB bus instead of a computer's other I/O ports.

RS232 MOD - RS232 Interface Module — TBA
Allows the DAM SYSTEMS MODULES to be used with an RS-232 port or terminal.

XPANDR1 - Expander Module — TBA
Allows up to 128 8-bit analog inputs (8 AIM16 Modules) to be connected to one system.

DAM SYSTEMS sets

AIM161 Starter Set — $189.00
Includes one AIM161, one POW1, one ICON and one OCON.

AIM162 Starter Set — $259.00
Includes one AIM162, one POW1, one ICON and one OCON.

PETSET1a — $295.00
Includes one PETMOD, one CABLE A24, one AIM161, one POW1 and one MANMOD1.

KIMSET1a — $285.00
Includes one KIMMOD, one CABLE A24, one AIM161, one POW1 and one MANMOD1.

NEXT IN ~~THIS~~ ISSUE

While this space is usually used to discuss the contents of the current issue, I would like to use it this month to talk about the exciting new changes coming up in the June 1979 issue of MICRO. These changes reflect our continuing effort to make MICRO even better than before.

The most significant change is that MICRO will be increased in size from the current 52 pages up to 68 pages. This is due to the continuing growth of both the articles submitted for publication and the increased interest in advertising in MICRO. The 16 page expansion will support growth in both of these areas.

The second most important change is that MICRO is going to be printed by a more sophisticated printing method. It will be printed on glossy stock which make for easier-to-read text, permits far superior halftones, and is slightly lighter so that mailing costs will remain about the same even though the size has increased.

One objection I have had to the current format of MICRO, an objection that has also been voiced by others, is that while the articles are the important part of MICRO, the overall magazine is a bit heavy or dry. To overcome this, some of the new space will be used for news, informal discussions, points-of-view, and so forth. I do not plan to publish "love-letters", but if you have something to say that may not merit an entire article - then write a short note. We will make room for these less formal presentations.

The overall appearance of MICRO will be improved - from the two color cover to the interior layout. We have analysed a number of other magazines and tried to "lift" those features that made them interesting and readable. I know that there are some "purists" in the audience who will object to any changes in the magazine, but I feel that most readers will appreciate the improvements. Some of the current features that we will definitely maintain are: the three-hole punch, the organization of each article into contiguous pages generally unbroken by ads, the protective mailing cover, and, of course, the editorial direction toward useful features and articles over games and "blue-sky" speculation.

With the increase in size and production cost, there will be an increase in price - but not that much. The retail price will increase to $2.00, but the subscription will only increase to $1.25 or $15.00 per year in the US. This is the first increase in price since we began 12 issues ago. Subscriptions will be accepted at the old rate until June 1, 1979 - so you may want to renew ahead (but only for one year).

MICROBES

EKIM or MAXI-KIM, MICRO 11:20

```
17D1 B0 AD  BCS START  should have been
17D1 B0 B4  BCS GETK
```

Robert A. Stein, Jr. reports that the table of memory size changes in "A CASSETTE OPERATING SYSTEM FOR THE APPLE II", MICRO 11:21 has some errors. The corrected table appears below:

If using CASSOS in other than a 16K machine change location $0358 as follows:

1F-8K 2F-12K 3F-16K 4F-20K 5F-24K 7F-32K 8F-36K BF-48K

CLUB ANNOUNCEMENTS

APPLESEED
c/o The Computer Shop
6812 San Pedro
San Antonio, TX 78216

(No information was included on their current meeting dates, nor was there a phone number given. This info would make the announcement much more useful !!)

An attempt is being made to organize an Apple group in New Hampshire. If you are interested, please contact:

Steve Adams
Governor Weare Apts.
Bldg. 1, Apt. 2
Seabrook, NH 03874
603/474-2230

ACG of NJ 6502/6800 User Group

Lew Edwards reports that the group is very active. "Meetings on 4th Friday at Union County Technical Institute have all kinds of expanded KIM's, PET's, an Apple group as well as AIM's and SYM's starting to show up. It's a wonderful way for beginners to get help from others in solving problems, getting their systems up and running, etc. Has really been taking off the last 6-7 months."

ABACUS (Apple Bay Area Computer Users Society)
Hayward BYTE Shop
1122 B Street
Hayward, CA

David R. Wilkerson, Secretary writes: "We have an active membership of 40, and we have developed a club library of 200+ programs. Currently we are negotiating to trade libraries with several other clubs." For more info call:

Ed Avelar, President
415/583-2431

Northwest Suburban Apple II Users Group

"Serving Apple II users in the Northwest Suburban Chicago area, we provide a forum for the interchange of knowledge, problems and application of the Apple II computer. Meetings are held the first Saturday of each month at the Palatine, Illinois Park District facility."

For more information please contact:

Ken Rose
650 Pompano Lane
Palatine, IL 60067
312/359-6723

ATTENTION ALL 6502 CLUBS

MICRO will be happy to donate a free six month subscription to any legitimate 6502 oriented club or user group. There are only two requirements for this offer:

1. A copy of the club/group mailing list must be sent to MICRO. This both shows that you are a real club and lets MICRO send a flyer to your members describing our publication.
2. Regular notification of meetings and events must be provided for this column. This will help us inform more potential members about your organization.

***** AIM-65 *****

P/N		Qty 1-9
A65-1	AIM-65 w/1K RAM	$375
A65-4	AIM-65 w/4K RAM	$450
A65-A	Assembler ROM	$85
A65-B	BASIC ROMS	$100

ACCESSORIES

P/N		
PRS1	+5V at 5A, +24V at 2.5A ±12V at 1A (does not fit inside ENC1)	$95
PRS2	+5V at 5A, +24V at 1A (mounts inside ENC1)	50
ENC1	AIM-65 case w/space for PRS2 and MEB1	45
MEB1	Memory expansion bd w/8K RAM; 8K PROM sockets and programmer for 2716; 6522 I/O chip	245
MEB2	Memory expansion bd w/16K RAM populated w/2114's	325
	Unpopulated	125
VIB1	Video bd w/128 char, 128 user char, prog. up to 100 char/line, up to 4K RAM, light pen interface and ASCII kybd interface	245
Thermal Paper Tape, 9/85' rolls		10

SYSTEMS

All AIM-65 systems are assembled and tested.

"A" series have the power supply external (PRS1).

"B" series have the power supply mounted inside (PRS2).

P/N		"A"	"B"
"STARTER" SYSTEMS			
S_65-1	A65-1 in ENC1	$495	$475
S_65-1B	Same Plus BASIC	595	575
S_65-4	A65-4 in ENC1	560	540
S_65-4B	Same Plus BASIC	660	640
"EXPANDED" SYSTEMS			
E_65-1	A65-1, ENC1, MEB1	$730	$710
E_65-1B	Same Plus BASIC	830	810
E_65-4	A65-4, ENC1, MEB1	795	775
E_65-4B	Same Plus BASIC	895	875

Higher quantities and systems with other options quoted upon request!

Mail Check or Money Order:

EXCERT, INCORPORATED
Attn: Laurie
4434 Thomas Avenue South
Minneapolis, Minnesota 55410
(612) 920-7792

Add $5.00 for shipping, insurance, and handling.

Minnesota residents add 4% sales tax.

AN AIM 65 USER's NOTES

Joe Burnett
16492 E. Tennessee Avenue
Aurora, CO 80012

The AIM 65 Microcomputer, made by Rockwell, is one of the newest, most versatile home computers available today. At the time of this writing (January 1979), it sells for $375. For this you get the complete computer, with a 20 character alphanumeric display, full size alphanumeric keyboard, a printer which uses inexpensive calculator type paper, 1K of RAM and 8K ROM-resident programming. Options include the ability to add 3K more memory, a 4K assembler, and an 8K Basic interpreter, all on-board, simply by purchasing them and plugging them in. An "application" connector and an "expansion" connector accept standard 44 pin edge connectors, and allow the control and I/O of two cassette units and a teletype, as well as off-board additional memory. On-board programming (ROM-resident) gives you the ability to display memory in either hex or mnemonic, alter memory, edit programming, turn the printer on and off, display registers, and enter any of the many resident subroutines. With cassette units connected, you can read or write to either one, and set up the AIM 65 to handle KIM-1 format (X1 or X3) or the AIM 65 format software. The AIM 65 will file and search cassette tapes, and the front panel alphanumeric display lets you know the status of the operation in progress as well as the block of data being read or written. Three keys on the keyboard (F1, F2, and F3) enable user defined functions through programmed jump instructions, and are a nice feature. Physically, the computer circuit board itself is ten inches deep by twelve inches wide, and the keyboard (which attaches through a supplied ribbon cable) is four inches deep by twelve inches wide. Included with the computer is a roll of paper for the printer, "feet" for the computer circuit board and the keyboard circuit, a User's Guide manual, an R6500 Programming manual, a System Hardware manual, a Programming Reference Card, an AIM 65 Summary Card, and a large schematic diagram, as well as the warranty card (don't forget to mail this in).

Software Compatibility

As with any new product, there are some problems. One is with the KIM-1 software. The KIM-1 is a very basic computer, and the AIM 65 is sophisticated by comparison. An example of the problem with the software is the KIM-1 "PLEASE" program. "PLEASE" loads data into memory locations which either are dedicated for use by the AIM 65, or are not present in the AIM 65. Consequently, although the AIM 65 can be initialized to accept KIM-1 programming, check the listing before you try to do it. It'll save you a lot of time and frustration. The AIM 65 User's Guide Manual includes a detailed memory map which you can use to determine (from a program listing) whether or not the program you're trying to load will in fact load as advertised.

Some Cassette Control Problems

A second problem is with the cassette unit control circuitry. There are actually two circuits in the AIM 65 for each cassette unit, and although Rockwell made an attempt to cover all eventualities, they didn't succeed. The first circuit makes use of an integrated circuit relay driver, which puts a low (ground) at the cassette control output pin of the "application" connector when the computer toggles the cassette unit "on". The second circuit is a transistor switch which is biased on when the computer toggles the cassette unit "on". The problem arises in that not all cassette units use a positive supply voltage with the negative line common (connected to the cassette unit frame). General Electric, for example, typically connects the positive side of the battery (or AC adapter) to the cassette unit frame, and uses negative voltage for the motor and electronic circuitry. At first glance, this doesn't look like a problem; after all, you only need to supply a closure to the remote switch line, and the cassette unit will run, right? Well, not quite. If you connect your GE cassette unit to the relay driver output pin, and the computer control has the cassette unit toggled "off", the cassette unit won't shut off. This is because you've put a negative voltage (from the cassette unit) at a point which has a nearly equal positive voltage (from the AIM 65), and the result is close enough to zero volts that the cassette unit motor runs even though the computer indicated that an "off" condition exists. Okay, so what about the transistor switch? Figure 9-4 of the User's Guide manual shows how to connect the wires. And the cassette unit won't run. At this point you're most likely very annoyed and confused (I know I was). The reason that the computer won't control the cassette unit is that (1) figure 9-4 of the User's Guide Manual is in error; the positive voltage from the cassette unit battery should go to pin "F", and the motor line should go to pin "E", of the "application" connector; and (2) the transistor does not have the voltages necessary to make it work, even after the wires are properly connected. If you look at the schematic diagram, you'll see that the transistor switch in the computer gets its operating voltage from the circuit it's controlling. To make it work, the transistor must have the proper bias (voltage between base and omitter), and to get this a common ground must exist between the computer power supply and the cassette unit power supply. It would seem that all that would be necessary would be to connect the emitter of the transistor (pin "F" of the "application" connector) to ground. Now the cassette unit will run and stop in response to computer control—until you plug in the ear and/or mic lines. When you do this, and the transistor turns on, you create a short circuit across the battery (or AC adapter) of the cassette unit. The reason is that when you wired up the ear/mic lines, you connected one side to ground on the 44 pin edge connector, and now the current finds a path through the cassette electronic circuitry, and everything stops. Under normal conditions, the remote switch on the cassette unit microphone is isolated from everything, so no problem exists. When you make the return line to the remote switch and the ear/mic line return common, a short circuit occurs. So what do you do now? Simulate an isolated switch, similar to what the microphone has. A relay is the only way, if you're going to control the cassette unit with the computer. Since my AIM 65 is still in the warranty period, I have not modified it as I'd like to. However, once the warranty period expires, I'm going to install two relays on the circuit board and use the transistor switches to control them. Then it won't matter what kind of motor control the cassette unit uses; I'll have the isolated switch action required to control any cassette unit, regardless of the polarity of the voltages involved.

At the time of this writing, neither the Assembler nor the BASIC interpreter is available from my distributor. This means that any programming I do has to be done using mnemonic codes. Although the documentation in the User's Guide is very good, the sample programs shown appear to have been produced with the use of an Assembler. An example is on pages 7-82 and 7-83. This program is intended to display and print an assembled message, but the information on how to prepare the message for storage in memory is absent. So, if you input this program you'll be " all dressed up with nowhere to go". The program shown below will allow you to input a message, and then retrieve it, all with the "bare bones" (1K RAM) AIM 65. How you use this is up to you. It could be just "for show", or you can modify it as desired and include it in more complex routines involving user interaction with the computer. This program does feature single key access (user function key F1, F2, or F3). Key F1 allows you to write to memory; key F2 retrieves the entire message; and key F3 retrieves the message a line at a time, with the space bar being used to advance the display to the next line of the message. The maximum length of the message is 13½ lines. An asterisk is typed at the end of the message when it is written to memory, which takes the computer out of the loop in all of the modes.

I hope the information in this article helps you avoid some of the problems and frustrations I've experienced. Enjoy your AIM 65. I'm having a lot of fun with mine, and I'm still learning what it's capabilities are.

```
                      WRITE TO MEMORY PROGRAM
                      JOE BURNETT
                      WITH MODS BY MIKE ROWE
                      APRIL 1979

0000                           ORG    $0000

                      AIM SUBROUTINES

0000                  CRCK   *      $EA24  DUMP PRINT BUFFER
0000                  CRLF   *      $E9F0  CARRIAGE RETURN/LINE FEED
0000                  INALL  *      $E993  INPUT FROM ANY DEVICE
0000                  OUTALL *      $E9BC  OUTPUT TO ANY DEVICE

                      ASCII CHARACTER

0000                  SPACE  *      $0020  SPACE CHARACTER
0000                  ASTER  *      $002A  ASTERISK CHARACTER

                      WRITE MESSAGE TO MEMORY

0000 20 F0 E9  WRITE   JSR    CRLF   CLEAR DISPLAY
0003 A0 00             LDYIM  $00    INIT MEMORY POINTER
0005 A2 13     LINE    LDXIM  $13    INIT CHARACTER COUNTER
0007 20 93 E9  INPUT   JSR    INALL  GET AN INPUT CHARACTER
000A 99 00 02          STAY   $0200  STORE IN BUFFER
000D C9 2A             CMPIM  ASTER  TEST TERMINATOR
000F F0 47             BEQ    EXIT   IF YES, THEN DONE
0011 C8                INY           BUMP POINTER
0012 CA                DEX           DECR CHARACTER COUNTER
0013 D0 F2             BNE    INPUT  IF NOT ZERO, GET MORE
0015 20 24 EA          JSR    CRCK   LINE FULL, SO PRINT IT
0018 4C 05 00          JMP    LINE   GET NEXT LINE

                      READ ENTIRE MESSAGE

001B 20 F0 E9  REM     JSR    CRLF   CLEAR DISPLAY
001E A0 00             LDYIM  $00    INIT MEMORY POINTER
0020 A2 13     RLINE   LDXIM  $13    INIT CHARACTER COUNTER
0022 B9 00 02  RCHAR   LDAY   $0200  GET CHARACTER FROM MEMORY
0025 C9 2A             CMPIM  ASTER  TEST FOR TERMINATOR
0027 F0 2F             BEQ    EXIT   IF YES, THEN DONE
0029 20 BC E9          JSR    OUTALL ELSE, DISPLAY CHARACTER
002C C8                INY           BUMP MEMORY POINTER
002D CA                DEX           DECR. CHARACTER COUNTER
002E D0 F2             BNE    RCHAR  IF NOT ZERO, GET NEXT CHARACTER
0030 20 24 EA          JSR    CRCK   ELSE, PRINT LINE
0033 4C 20 00          JMP    RLINE  THEN CONTINUE
```

READ MESSAGE ONE LINE AT A TIME

```
0036 20 F0 E9  ONELIN JSR   CRLF   CLEAR DISPLAY
0039 A0 00            LDYIM $00    INIT MEMORY POINTER
003B A2 13     OLINE  LDXIM $13    INIT CHARACTER COUNTER
003D B9 00 02  OCHAR  LDAY  $0200  GET CHARACTER FROM MEMORY
0040 C9 2A            CMPIM ASTER  TEST TERMINATOR
0042 F0 14            BEQ   EXIT   IF YES, THEN DONE
0044 20 BC E9         JSR   OUTALL ELSE, PRINT CHARACTER
0047 C8               INY          BUMP MEMORY POINTER
0048 CA               DEX          DECR CHARACTER COUNTER
0049 D0 F2            BNE   OCHAR  IF NOT ZERO, CONTINUE
004B 20 93 E9  WAIT   JSR   INALL  ELSE WAIT FOR A SPACE
004E C9 20            CMPIM SPACE  FROM KAYBOARD TO CONTINUE
0050 D0 F9            BNE   WAIT   NOT A SPACE
0052 20 24 EA         JSR   CRCK   SPACE, SO PRINT
0055 4C 3B 00         JMP   OLINE  THEN GET NEXT LINE
```

COMMON EXIT ROUTINE TO CLEAN UP
THE DISPLAY AND RETURN TO MONITOR

```
0058 20 F0 E9  EXIT   JSR   CRLF   OUTPUT TO BLANK LINES
005B 20 F0 E9         JSR   CRLF
005E 00               BRK          THEN EXIT TO MONITOR
```

USER FUNCTION DEFINITIONS

```
010C                  ORG   $010C

010C 4C 00 00         JMP   WRITE  F1 TO WRITE MESSAGE
010F 4C 1B 00         JMP   REM    F2 TO READ ENTIRE MESSAGE
0112 4C 36 00         JMP   ONELIN F3 TO READ ONE LINE AT A TIME
```

```
<K>*=0
/FF
 0000 20 JSR E9F0
 0003 A0 LDY #00
 0005 A2 LDX #13
 0007 20 JSR E993
 000A 99 STA 0200,Y
 000D C9 CMP #2A
 000F F0 BEQ 0058
 0011 C8 INY
 0012 CA DEX
 0013 D0 BNE 0007
 0015 20 JSR EA24
 0018 4C JMP 0005
 001B 20 JSR E9F0
 001E A0 LDY #00
 0020 A2 LDX #13
 0022 B9 LDA 0200,Y
 0025 C9 CMP #2A
 0027 F0 BEQ 0058
 0029 20 JSR E9BC
 002C C8 INY
 002D CA DEX
 002E D0 BNE 0022
 0030 20 JSR EA24
 0033 4C JMP 0020
 0036 20 JSR E9F0
 0039 A0 LDY #00
 003B A2 LDX #13
 003D B9 LDA 0200,Y
 0040 C9 CMP #2A
 0042 F0 BEQ 0058
 0044 20 JSR E9BC
 0047 C8 INY
 0048 CA DEX
 0049 D0 BNE 003D
 004B 20 JSR E993
 004E C9 CMP #20
 0050 D0 BNE 004B
 0052 20 JSR EA24
 0055 4C JMP 003B
 0058 20 JSR E9F0
 005B 20 JSR E9F0
 005E 00 BRK
<K>*=10C
/3?
 010C 4C JMP 0000
 010F 4C JMP 001B
 0112 4C JMP 0036
```

APPLE II® PROFESSIONAL SOFTWARE

PIE TEXT EDITOR

PIE (PROGRAMMA IMPROVED EDITOR) is a two-dimensional cursor-based editor designed specifically for use with memory-mapped and cursor-based CRT's. It is totally different from the usual line-based editors, which were originally designed for Teletypes. The keys of the system input keyboard are assigned specific PIE Editor function commands. Some of the features included in the PIE system are: Blinking Cursor; Cursor movement up, down, right, left, plus tabs; Character insert and delete; String search forwards and backwards; Page scrolling; GOTO line number, plus top or bottom of file; Line Insert and delete anywhere on screen; Move and copy (single and multiple lines); Append and clear to end of line; Efficient memory usage. The following commands are available in the PIE Text Editor and each is executed by depressing the systems argument key simultaneously with the command key desired:

[LEFT]	Move cursor one position to the left
[RGHT]	Move cursor one position to the right
[UP]	Move cursor up one line
[DOWN]	Move cursor down one line
[BHOM]	Home cursor in lower left left hand corner
[HOME]	Home cursor in upper left hand corner
[-PAG]	Move up (toward top of file) one "page"
[+PAG]	Move down (toward bottom of file) one "page"
[LTAB]	Move cursor left one horizontal tab
[RTAB]	Move cursor right one horizontal tab
[GOTO]	Go to top of file (line 1)
[ARG]n[GOTO]	Go to line 'n'
[BOT]	Go to bottom of file (last line + 1)
[-SCH]	Search backwards (up) into file for the next occurence of the string specified in the last search command
[ARG]t[-SCH]	Search beckwards for string 't'
[+SCH]	Search forwards (down) into the file for the next occurence of the string specified in the last search command
[ARG]t[+SCH]	Search forward for string 't'
[APP]	Append -move cursor to last character of line +1
[INS]	Insert a blank line beforere the current line
[ARG]n[INS]	Insert 'n' blank lines before the current line
[DEL]	Delete the current line, saving it in the "push" buffer
[ARG]n[DEL]	Delete 'n' lines and save the first 20 in the "push" buffer
[DBLK]	Delete the current line as long as it is blank
[PUSH]	Save current line in "push" buffer
[ARG]n[PUSH]	Save 'n' lines in the "push" buffer
[POP]	Copy the contents of the "push" buffer before the current line
[CINS]	Enable character insert mode
[CINS][CINS]	Turn off character insert mode
[BS]	Backspace
[GOB]	Gobble - delete the current character and pull remainder of characters to right of cursor left one position
[EXIT]	Scroll all text off the screen and exit the editor
[ARG][HOME]	Home Line - scroll up to move current line to top of screen
[APP][APP]	Left justify cursor on current line
[ARG][GOB]	Clear to end of line

Apple PIE Cassette	**16K**	**$19.95**
TRS-80PIE Cassette	**16K**	**19.95**
Apple PIE Disk	**32K**	**24.95**

6502FORTH · Z-80FORTH 6800 FORTH

FORTH is a unique threaded language that is ideally suited for systems and applications programming on a micro-processor system. The user may have the interactive FORTH Compiler/Interpreter system running stand-alone in 8K to 12K bytes of RAM. The system also offers a built-in incremental assembler and text editor. Since the FORTH language is vocabulary based, the user may tailor the system to resemble the needs and structure of any specific application. Programming in FORTH consists of defining new words, which draw upon the existing vocabulary, and which in turn may be used to define even more complex applications. Reverse Polish Notation and LIFO stacks are used in the FORTH system to process arithmetic expressions. Programs written in FORTH are compact and very fast.

SYSTEM FEATURES & FACILITIES

Standard Vocabulary with 200 words
Incremental Assembler
Structured Programming Constructs
Text Editor
Block 1/0 Buffers
Cassette Based System
User Defined Stacks
Variable Length Stacks
User Defined Dictionary
Logical Dictionary Limit
Error Detection
Buffered Input

CONFIGURATIONS

AppleFORTH Cassette 16K	**$34.95**
AppleFORTH Disk 32K	**49.95**
PetFORTH Cassette 16K	**34.95**
TRS-80FORTH Cassette 16K	**34.95**
SWTPCFORTH Cassette 16K	**34.95**

ASM/65 EDITOR ASSEMBLER

ASM/65 is a powerful, 2 pass disk-based assembler for the Apple II Computer System. It is a compatible subset of the FORTRAN cross-assemblers which are available for the 6500 family of micro-processors. ASM/65 features many powerful capabilities, which are under direct control of the user. The PIE Text Editor co-resides with the ASM/65 Assembler to form a comprehensive development tool for the assembler language programmer. Following are some of the features available in the ASM/65 Editor Assembler.

- PIE Text Editor Command Repetoire
- Disk Based System
- Decimal, Hexadecimal, Octal, & Binary Constants
- ASCII Literal Constants
- One to Six character long symbols
- Location counter addressing "*"
- Addition & Subtraction Operators in Expressions
- High-Byte Selection Operator
- Low-Byte Selection Operator
- Source statements of the form: [label] [opcode] [operand] [;comment]
- 56 valid machine instruction mnemonics
- All valid addressing modes
- Equate Directive
- BYTE Directive to initialize memory locations
- WORD Directive to initialize 16-bit words
- PAGE Directive to control source listing
- SKIP Directive to control source listing
- OPT Directive to set select options
- LINK Directive to chain multiple text files
- Comments
- Source listing with object code and source statements
- Sorted symbol table listing

CONFIGURATION

Apple II	**48K/Disk**	**$69.95**

LISA INTERACTIVE ASSEMBLER

LISA is a totally new concept in assembly language programming. Whereas all other assemblers use a separate or co-resident text editor to enter the assembly language program and then an assembler to assemble the source code, LISA is fully interactive and performs syntax/addressing mode checks as the source code is entered in. This is similar in operation to the Apple II Integer BASIC Interpreter. All error messages that are displayed are in plain, easy to understand English, and not simply an Error Code. Commands in LISA are structured as close as possible to those in BASIC. Commands that are included are: LIST, DELETE, INSERT, PR #n, IN#n, SAVE, LOAD, APPEND, ASM, and a special user-defineable key envisioned for use with "dumb" peripherals. LISA is DISK II based and will assemble programs with a textfile too long to fit into the Apple memory. Likewise, the code generated can also be stored on the Disk, hence freeing up memory for even larger source programs. Despite these Disk features, LISA is very fast; in fact LISA is faster than most other commercially available assemblers for the Apple II. Not only is LISA faster, but also, due to code compression techniques used LISA requires less memory space for the text file. A full source listing containing the object and source code are produced by LISA, in addition to the symbol table

Apple II 32K/Disk $34.95

PROGRAMMA INTERNATIONAL, INC.

3400 Wilshire Blvd.
Los Angeles, CA 90010

(213) 384-0579 • 384-1116 • 384-1117

Apple II is a registered trademark of Apple Computers, Inc. These professional products are available at your local computer dealer.

PROGRAMMA Software Products

S-C ASSEMBLER II
Super Apple II Assembler

Chuck Carpenter
2228 Montclair Pl.
Carrollton, TX 75006

I've had the good fortune to get an advance copy of an excellent assembler for the Apple II. The assembler was written by Bob Sander-Cederlof and has many desireable features. Bob has used sweet 16 and several routines from the monitor and integar BASIC (it doesn't run with the Applesoft ROM on). The result is a compact co-resident two-pass assembler. A summary of assembler commands and data is listed in Table 1.

Here are a few of the assembler features:

- Format compatible with Apple mini-assembler
- Complete text editing using standard Apple screen and line editing features.
- Save and Load as in integar BASIC
- Psuedo op codes
- Text for REMs following the line no.
- Tabs to the opcode, operand and comment field using (CTRL) I
- Symbol table
- Listing, fast or slow
- Stop and start a LIST or ASM at any time
- Access Apple monitor from the assembler using $
- Run programs from the assembler

The S-C ASSEMBLER II includes many other features. Among these are:

- Line renumbering starting at 1000 by 10's
- Printer driver routine - his or yours (or mine for that matter).
- Pagination of printed output
- Program location and relocation
- Can be used to renumber BASIC programs (except branches)
- Operates within DOS (see Table 2)
- Runs on an 8K machine

I have included a couple of examples of the S-C ASSEMBLER II features in Figure 1 and 2. Figure 1 is a functional routine. Figure 2 is merely for illustration of the .DA feature. Most of the assembler capability is illustrated in Figure 1. This routine, which compares 2 byte data, can be used for many applications such as extended loop counters. The example also includes ASCII strings using the pseudo op code .AS.

A jump to the user exit at $3F8 was used to enter the data. This also takes advantage of the (CTRL) Y feature of the Apple monitor.

By calling the print routine with PRT, a hard copy of a listing or of assembled output is obtained. The printer driver routine is output from the game paddle connector. This is a TTL level serial signal. Typing SLO(W) or FAS(T) stops the printer output. Also, SLO(W) will provide a slow listing of your program. You can stop and start the listing with the space bar and, escape back to the assembler with a (RETURN). FAS(T) cancels SLO(W) returning to normal screen speed. (See Slow List, MICRO #5 page 21.)

For text editing, you can insert a line between other lines and list any single line or combination of lines. This allows character editing or line editing using Apple ESCAPE functions ((ESCAPE)D,C,B). Also you can DEL(ETE) any line or combination of lines.

An asterisk (*) in the first column of the label field allows that line to be a comment or blank line. Very useful for commenting a program. I used short comments in my programs; I only have 48 columns. Actually the comment can be any length (up to 100 characters or so). An asterisk used in the operand field means current location. You can add or subtract labels, hex and decimal values from the current location. Each of these can be added or subtracted, to or from, each other. Here are some examples:

```
1000 LABL LDA *-*  CURRENT-CURRENT
1010 LAB2 LDA LABL-LABL
1020 LAB3 LDA *-LABL
1030 LAB4 LDA LABL+1234
1040 LAB5 LDA $1234-LABL
1050 LAB6 LDA $ABCD-5678
1060 *
1070 * EXAMPLES OF ADDITION & SUBTRACTION OF
1080 * CURRENT VALUE, LABELS, DECIMAL AND
1090 * HEX VALUES FROM EACH OTHER.
1100 *
```

Illustration of the .DA feature is shown in Figure 2. The intent here is to show data in a single or 2 byte location. Once the data value has been assigned with the .DA code, it can be manipulated with another feature. This feature is shown as a / (slant line) and # (pound) in the first column of the operand field. Here's what's happening:

LDA /LABL = HIBYTE = ÷256
LDA #LABL = LOBYTE = MOD256

As you can see from this and the previous examples, these features provide a very powerful assembler capability.

Before I obtained this assembler I could never get very enthusiastic about extensive machine or assembly language programming. Now, with this assembler, this coding is as easy as BASIC. You can get a copy for your Apple II from:

S-C SOFTWARE
P.O. Box 5537
Richardson, TX 75080
Price - $25.00

I think you will enjoy it: having the efficiency of machine language programs developed with the ease of BASIC. The combination of compact programs with interactive capability makes personal computing even more enjoyable.

Load:	*1000.1CFFR
Run:	*1000G Hard Entry
or:	*1003G Soft Entry

Pseudo ops:

label .OR expr	origin (optional label)
label .EQ expr	equate
label .DA expr	data (optional label)
label .HS xxxx...x	hex string
label .AS daaaa...ad	ascii string (d is any delimiter)
.EN	end

Commands:

LOAD	load program from tape
SAVE	save program to tape
LIST	list entire program
LIST line#	list selected line
LIST line#,line#	list range of lines
DELETE line#	delete selected line
DELETE line#,line#	delete range of lines
RENUMBER	renumbers all lines
NEW	erase program
SLOW	program slow list
FAST	program fast list
PRT	printer driver $1B77-1BFF
ASM	assemble program
RUN expr	execute starting at expr
APPEND	add program from tape to one in memory

Table 1
S-C Assembler II Summary Notes

Instruction Steps:

1. Bring up DOS per instruction manual
2. Reset to monitor (*)
3. Load assembler from tape
4. Return to DOS using $3DOG
5. BSAVE Assembler
6. LOCK Assembler
7. Call 4096 Jumps to Assembler
8. $3DOG Jumps to DOS soft entry but...

At this point the DOS is clobbered. Any further use of DOS requires a reboot. It is very handy though to have the speed of loading the assembler from the disc.

Table 2
S-C Assembler II with Apple II DOS

```
:ASM                1000 * .DA PSEUDO OP EXAMPLE
                    1010 *
                    1020        .OR $300
0300- 34 12         1030 HEX   .DA $1234
0302- 34 12         1040 DEC   .DA 4660
                    1050 *
                    1060 * ADDRESS OF DATA
                    1070 *
0304- A9 00         1080        LDA #HEX      HEX LO BYTE
0306- A9 03         1090        LDA /HEX      HEX HI BYTE
                    1100 *
                    1110 * DATA AT THE ADDRESS
                    1120 *
0308- AD 02 03      1130        LDA DEC       DEC LO BYTE
030B- AD 03 03      1140        LDA DEC+1     DEC HI BYTE

                    1150        .EN

SYMBOL TABLE

HEX   0300     DEC   0302

:
```

Figure 2
DA Pseudo Op Example

```
:NEW
S-C ASSEMBLER ][

:LOAD

:ASM
                              1000 * S-C ASSEMBLER II EXAMPLE
                              1010 *
                              1020 * COMPARES HEX VALUES
                              1030 * AND INDICATES WHICH
                              1040 * IS GREATER (OR EQUAL).
                              1050 *
                              1060        .OR $300
                              1070 *
                              1080 *  .OR DEFAULT IS $0800
                              1090 *
                              1100 COUT .EQ $FDED
              0300- 58 20 3C
              0303- 20 59     1110 LESS .AS 'X < Y'
              0305- 8D        1120        .HS 8D
              0306- 58 20 3E
              0309- 3D 20 59  1130 GREQ .AS 'X >= Y'
              030C- 8D        1140        .HS 8D
                              1150 XL   .EQ $3C
                              1160 XH   .EQ $3D
                              1170 YL   .EQ $3E
                              1180 YH   .EQ $3F
              030D- A5 3C     1190 STAR LDA XL
              030F- C5 3E     1200      CMP YL
              0311- A5 3D     1210      LDA XH
              0313- E5 3F     1220      SBC YH
              0315- B0 06     1230      BCS TST1     X >= Y
              0317- A0 00     1240      LDY #LESS-LESS
              0319- 20 28 03  1250      JSR PRNT
              031C- 60        1260      RTS
              031D- A0 06     1270 TST1 LDY #GREQ-LESS
              031F- 4C 28 03  1280      JMP PRNT
              0322- 09 80     1290 PRT1 ORA #$80     NORMAL OUT
              0324- 20 ED FD  1300      JSR COUT
              0327- C8        1310      INY
              0328- B9 00 03  1320 PRNT LDA LESS,Y
              032B- 10 F5     1330      BPL PRT1
              032D- 4C ED FD  1340      JMP COUT
                              1350 *
                              1360 * DATA ENTRY THROUGH
                              1370 * USER EXIT @ $3F8.
                              1380 *
                              1390 * DATA.DATA (CTRL)Y
                              1400 *
                              1410      .OR $3F8
              03F8- 4C 0D 03  1420      JMP STAR
                              1430      .EN

              SYMBOL TABLE

              COUT  FDED    LESS  0300    GREQ  0306
              XL    003C    XH    003D    YL    003E
              YH    003F    STAR  030D    TST1  031D
              PRT1  0322    PRNT  0328

              :
```

```
:
:EXAMPLE RUN
:
:$10000.20000
X >= Y

:$2000.1000
X >= Y

:$3.3
X >= Y

:
```

Figure 1
S-C Assembler II Example

A PET HEX DUMP PROGRAM

Joseph Donato
193 Walford Rd. E.
Sudbury, ONT., Canada

Have you PET owners ever wondered how it could be possible to look at your BASIC which resides in Read Only Memory (ROM)? To be able to look for routines entry points and other interesting codes in machine language?

This program will do just that. You can look at all memory locations in PET's BASIC which starts at 49152 decimal or COOO hexadecimal in memory. One is able for example to look at locations D71E through D890 where addition and subtraction routines are carried out, D8BF through D8FC where the log function is evaluated, D9E1 through DA73 where division is performed and many other locations where other routines are carried out.

A start for this program was provided by Mr. Herman's article of MICRO 7:47. Of course the same information was available in the Commodore Users Notes.

In any event I decided that the ultimate goal of the program would be to provide a memory dump of some sort in hexadecimal notation so that machine language instructions could easily be recognized.

The output of the program is formatted as a starting address followed by either 32 or 8 bytes of data per line, all in hexadecimal, depending on whether or not a printer is to be used. With the data bytes in hex notation it is very easy to correlate them with the 6502 microprocessor machine language instruction set.

The program listing has been thoroughly debugged and tested. Although the program was originally written for a PET with a Centronics printer, as I outlined in the REM's, the program will run on a "bare" PET with no problem.

The changes for a "bare" PET are as follows:

1. Omit line 10.
2. Change line 542 to read:
   542 IF L<>9 THEN 570
3. Omit all print statements and substitute instead the print format outlined in the REM's at lines 606 through 612. These print lines are to be placed at line 545, 546, 547, 548.
4. Notice that there is no comma or semicolon after the last print character. This is very important otherwise the format will be destroyed.

A considerable amount of time was spent on both versions of the program. No problems were encountered in running either version.

I hope that by following the machine language coding of the 6502 some of you will obtain a better understanding of PET's Basic 'inner workings'. Also some of you who have the T.I.M. monitor will be able to trace its subroutines and jumps to Basic. Perhaps it may inspire you in writing some machine language programs or routines.

I should add that if one wishes to look at different addresses other than the COOO (49152 decimal), all you need do is to change the starting address value "K" in line 240. This must be in decimal notation

I hope you get as much pleasure as I did 'sneaking a look' at PET's Basic.

```
1 REM *** A BASIC PET HEX DUMP ***
2 REM THIS PROGRAM WILL PEEK AT PET'S
3 REM MEMORY IN ROM STARTING AT A GIVEN ADDRESS 'K' (49152 DECIMAL) AND RETURN
4 REM THE CORRESPONDING DATA.  ALL VALUES ARE CONVERTED TO HEXADECIMAL PRIOR TO
5 REM PRINTING.  THE FORMAT IS: STARTING ADDRESS PLUS 32 OR 8 BYTES OF DATA,
6 REM PER LINE DEPENDING WHETHER OR NOT A PRINTER IS USED.
7 REM
8 REM THE COMMAND ON LINE 10 INITIALIZES THE PRINTER PORT.  IT *MUST* BE OMITTED
9 REM IF A "BARE" PET IS USED.
10 OPEN 5,5:CMD 5
11 REM FOLLOWING IS A MACHINE LANGUAGE
12 REM ROUTINE WHICH RESIDES IN NUMBER 2 TAPE
13 REM BUFFER AREA.  IT RETURNS THE CONTENTS OF THE CORRESPONDING MEMORY
14 REM LOCATIONS SPECIFIED BY 'K'.
15 POKE(1),58
16 POKE(2),3
17 POKE(826),32
20 POKE(827),167
30 POKE(828),208
40 POKE(829),166
```

```
50 POKE(830),179
60 POKE(831),164
70 POKE(832),180
80 POKE(833),134
90 POKE(834),180
100 POKE(835),132
120 POKE(836),179
130 POKE(837),162
140 POKE(838),00
150 POKE(839),161
160 POKE(840),179
170 POKE(841),168
180 POKE(842),169
190 POKE(843),00
200 POKE(844),32
210 POKE(845),120
220 POKE(846),210
230 POKE(847),96
232 REM SET UP STORAGE AREA FOR ONE
233 REM LINE OF HEX VALUES TO BE PRINTED
235 DIM N1$(40),NO$(40)
236 REM INITIALIZE CHARACTER COUNTER
237 L=1
238 REM THE VALUE OF 'K' DETERMINES
239 REM THE STARTING ADDRESS.
240 FOR K=49152 TO 65536
241 I=K
250 A=USR(K-65536)
255 REM LINES 270-530 CONSIST OF A SUBROUTINE TO CONVERT ALL VALUES FROM
256 REM DECIMAL TO HEXADECIMAL NOTATION
270 B%=16
280 D=A
390 H$="0123456789ABCDEF"
400 NO$(L)=""
405 N1$(L)=""
410 F%=LOG(I)/LOG(B%)
411 REM BECAUSE THE DECIMAL TO HEX ROUTINE
412 REM RETURNS A SINGLE '0' FOR VALUES
413 REM OF A=0, LINE 416 CONVERTS
414 REM ANY OF THESE ZERO VALUES TO
415 REM A DOUBLE HEX '00'.
416 IF A=0 THEN NO$(L)="00":GOTO 480
418 G%=LOG(D)/LOG(B%)
420 FOR J=G% TO 0 STEP -1
430 X=INT(B%^J)
440 C%=D/X
445 REM LINE 455 INSERTS A LEADING ZERO
446 REM IN HEXADECIMAL VALUES OF LESS
447 REM THAN 'F'(15). EX. '7'='07' ETC.
450 NO$(L)=NO$(L)+MID$(H$,C%+1,1)
455 IF A<16 THEN NO$(L)=('0'+NO$(L))
460 D=INT(D-C%*X)
470 NEXT J
480 FOR J=F% TO 0 STEP -1
490 X=INT(B%^J)
500 C%=INT(I/X)
510 N1$(L)=N1$(L)+MID$(H$,C%+1,1)
520 I=INT(I-C%*X)
530 NEXT J
```

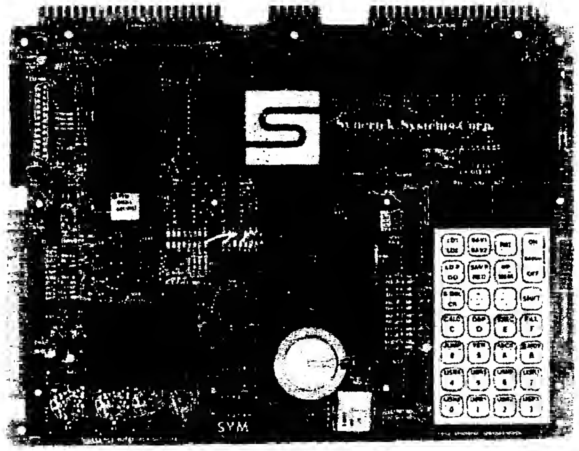
```
532 REM SUBROUTINE FOR DECIMAL TO HEXADECIMAL CONVERSION ENDS HERE
535 L=L+1
536 REM LINE 542 CHECKS TO SEE IF THE
537 REM REQUIRED NUMBER OF CHARACTERS
538 PER LINE HAVE BEEN DONE.  THE TEST VALUE
539 NUMBER 33 *MUST* BE CHANGED TO A NUMBER 9 IF A "BARE" PET IS USED.
542 IF L<>33 THEN 570
545 PRINT N1$(1),"  ",NO$(1),"  ",NO$(2),"  ",NO$(3),"  ",NO$(4),"  ",NO$(5),
546 PRINT "  ",NO$(6),"  ",NO$(7),"  ",NO$(8),"  ",NO$(9),"  ",NO$(10),"  ",
547 PRINT NO$(11),"  ",NO$(12),"  ",NO$(13),"  ",NO$(14),"  ",NO$(15),"  ",
548 PRINT NO$(16),"  ",NO$(17),"  ",NO$(18),"  ",NO$(19),"  ",NO$(20),"  ",
549 PRINT NO$(21),"  ",NO$(22),"  ",NO$(23),"  ",NO$(24),"  ",NO$(25),"  ",
550 PRINT NO$(26),"  ",NO$(27),"  ",NO$(28),"  ",NO$(29),"  ",NO$(30),"  ",
560 PRINT NO$(31),"  ",NO$(32)
565 L=1
570 NEXT K
600 REM THE PRINT STATEMENT FOR THE PET
602 REM WITH NO PRINTER "BARE" SHOULD BE AS FOLLOWS:
606 REM PRINT N1$(1);"  ";NO$(1),"  ";
608 REM NO$(2);"  ";NO$(3);"  "NO$(4);
610 REM "  ";NO$(5);"  ";NO$(6);"  ";
612 REM NO$(7);"  ";NO$(8);"  ";NO$(9)
615 END
```

```
C000 1D C7 48 C6 35 CC EF C7 C5 CA DF CA 70 CF 23 C8 9C C8 9C C7 74 C7 1F C8 0C C7 7F C7 C9 C7 32 C8
C020 1B C7 42 C8 01 D7 D4 FF D7 FF DA FF 94 D2 F8 D6 7E C9 9E C9 44 C7 A7 C5 6F C7 84 C9 DD FF BF FF
C040 C2 FF 9E CA 3A C5 8E D8 9E D8 2A D8 00 00 64 D2 85 D2 24 DE 45 DF BF D8 A0 DE 9E DF A5 DF EE DF
C060 48 E0 E6 D6 54 D6 49 D3 85 D6 63 D6 C4 D5 D8 D5 04 D6 0F D6 79 3E D7 79 27 D7 7B FF D8 7B E3 D9
C080 7F 20 DE 50 D8 CE 46 D5 CE 70 66 DE 5A E7 CD 64 05 CF 45 4E C4 46 4F D2 4E 45 58 D4 44 41 54 C1
C0A0 49 4E 50 55 54 A3 49 4E 50 55 D4 44 49 CD 52 45 41 C4 4C 45 D4 47 4F 54 CF 52 55 CE 49 C6 52 45
C0C0 53 54 4F 52 C5 47 4F 53 55 C2 52 45 54 55 52 CE 52 45 CD 53 54 4F D0 4F CE 57 41 49 D4 4C 4F 41
C0E0 C4 53 41 56 C5 56 45 52 49 46 D9 44 45 C6 50 4F 4B C5 50 52 49 4E 54 A3 50 52 49 4E D4 43 4F 4E
C100 D4 4C 49 53 D4 43 4C D2 43 4D C4 53 59 D3 4F 50 45 CE 43 4C 4F 53 C5 47 45 D4 4E 45 D7 54 41 42
C120 A8 54 CF 46 CE 53 50 43 A8 54 48 45 CE 4E 4F D4 53 54 45 D0 AB AD AA AF DE 41 4E C4 4F D2 BE BD
C140 BC 53 47 CE 49 4E D4 41 42 D3 55 53 D2 46 52 C5 50 4F D3 53 51 D2 52 4E C4 4C 4F C7 45 58 D0 43
C160 4F D3 53 49 CE 54 41 CE 41 54 CE 50 45 45 CB 4C 45 CE 53 54 52 A4 56 41 CC 41 53 C3 43 48 52 A4
C180 4C 45 46 54 A4 52 49 47 48 54 A4 4D 49 44 A4 00 4E 45 58 54 20 57 49 54 48 4F 55 54 20 46 4F D2
C1A0 53 59 4E 54 41 D8 52 45 54 55 52 4E 20 57 49 54 48 4F 55 54 20 47 4F 53 55 C2 4F 55 54 20 4F 46
C1C0 20 44 41 54 C1 49 4C 4C 45 47 41 4C 20 51 55 41 4E 54 49 54 D9 00 00 00 00 00 4F 56 45 52 46 4C
C1E0 4F D7 4F 55 54 20 4F 46 20 4D 45 4D 4F 52 D9 55 4E 44 45 46 27 44 20 53 54 41 54 45 4D 45 4E D4
C200 42 41 44 20 53 55 42 53 43 52 49 50 D4 52 45 44 49 4D 27 44 20 41 52 52 41 D9 44 49 56 49 53 49
C220 4F 4E 20 42 59 20 5A 45 52 CF 49 4C 4C 45 47 41 4C 20 44 49 52 45 43 D4 54 59 50 45 20 4D 49 53
C240 4D 41 54 43 C8 53 54 52 49 4E 47 20 54 4F 4F 20 4C 4F 4E C7 42 41 44 20 44 41 54 C1 46 4F 52 4D
C260 55 4C 41 20 54 4F 4F 20 43 4F 4D 50 4C 45 D8 43 41 4E 27 54 20 43 4F 4E 54 49 4E 55 C5 55 4E 44
C280 45 46 27 44 20 46 55 4E 43 54 49 4F CE 20 45 52 52 4F 52 00 20 49 4E 20 00 0D 0A 52 45 41 44 59
C2A0 2E 0D 0A 00 0D 0A 42 52 45 41 4B 00 BA E8 E8 E8 E8 BD 01 01 C9 81 D0 21 A5 99 D0 0A BD 02 01 85
C2C0 98 BD 03 01 85 99 DD 03 01 D0 07 A5 98 DD 02 01 F0 07 8A 18 69 12 AA D0 D8 60 20 2A C3 85 80 84
C2E0 81 38 A5 A9 E5 AE 85 71 A8 A5 AA E5 AF AA E8 98 F0 23 A5 A9 38 E5 71 85 A9 B0 03 C6 AA 38 A5 A7
C300 E5 71 85 A7 B0 08 C6 A8 90 04 B1 A9 91 A7 88 D0 F9 B1 A9 91 A7 C6 AA C6 A8 CA D0 F2 60 0A 69 36
C320 B0 35 85 71 BA E4 71 90 2E 60 C4 83 90 28 D0 04 C5 82 90 22 48 A2 09 98 48 B5 A6 CA 10 FA 20 04
C340 D4 A2 F7 68 95 B0 E8 30 FA 68 A8 68 C4 83 90 06 D0 05 C5 82 B0 01 60 A2 52 46 64 A5 03 F0 07 20
C360 CC FF A9 00 85 03 20 D2 C9 20 47 CA BD 90 C1 48 29 7F 20 49 CA E8 68 10 F3 20 84 C5 A9 8D A0 C2
C380 20 27 CA A4 89 C8 F0 03 20 94 DC 46 64 A9 99 A0 C2 20 27 CA 20 68 C4 86 C9 84 CA 20 C2 00 F0 F4
C3A0 A2 FF 86 89 90 06 20 8D C4 4C E9 C6 20 63 C8 20 8D C4 84 5C 20 22 C5 90 44 A0 01 B1 AE 85 72 A5

BREAK IN 240
READY.
```

Example of a partial Hex Dump obtained with the Program

SYM-1, 6502-BASED MICROCOMPUTER

- FULLY-ASSEMBLED AND COMPLETELY INTEGRATED SYSTEM that's ready-to-use
- ALL LSI IC'S ARE IN SOCKETS
- 28 DOUBLE-FUNCTION KEYPAD INCLUDING UP TO 24 "SPECIAL" FUNCTIONS
- EASY-TO-VIEW 6-DIGIT HEX LED DISPLAY
- KIM-1* HARDWARE COMPATIBILITY
  The powerful 6502 8-Bit MICROPROCESSOR whose advanced architectural features have made it one of the largest selling "micros" on the market today.
- THREE ON-BOARD PROGRAMMABLE INTERVAL TIMERS available to the user, expandable to five on-board.
- 4K BYTE ROM RESIDENT MONITOR and Operating Programs.
- Single 5 Volt power supply is all that is required.
- 1K BYTES OF 2114 STATIC RAM onboard with sockets provided for immediate expansion to 4K bytes onboard, with total memory expansion to 65, 536 bytes.
- USER PROM/ROM: The system is equipped with 3 PROM/ROM expansion sockets for 2316/2332 ROMs or 2716 EPROMs
- ENHANCED SOFTWARE with simplified user interface
- STANDARD INTERFACES INCLUDE:
  - —Audio Cassette Recorder Interface with Remote Control (Two modes: 135 Baud KIM-1* compatible, Hi-Speed 1500 Baud)
  - —Full duplex 20mA Teletype Interface
  - —System Expansion Bus Interface
  - —TV Controller Board Interface
  - —CRT Compatible Interface (RS-232)
- APPLICATION PORT: 15 Bi-directional TTL Lines for user applications with expansion capability for added lines
- EXPANSION PORT FOR ADD-ON MODULES (51 I/O Lines included in the basic system)
- SEPARATE POWER SUPPLY connector for easy disconnect of the d-c power
- AUDIBLE RESPONSE KEYPAD



Synertek has enhanced KIM-1* software as well as the hardware. The software has simplified the user interface. The basic SYM-1 system is programmed in machine language. Monitor status is easily accessible, and the monitor gives the keypad user the same full functional capability of the TTY user. The SYM-1 has everything the KIM-1* has to offer, plus so much more that we cannot begin to tell you here. So, if you want to know more, the SYM-1 User Manual is available, separately.

SYM-1 Complete w/manuals	**$269.00**
SYM-1 User Manual Only	**7.00**
SYM-1 Expansion Kit	**75.00**

Expansion includes 3K of 2114 RAM chips and 1-6522 I/O chip.

SYM-1 Manuals: The well organized documentation package is complete and easy-to-understand.

SYM-1 CAN GROW AS YOU GROW. Its the system to BUILD-ON. Expansion features that are soon to be offered:

***BAS-1 8K Basic ROM (Microsoft)**	**$159.00**
***KTM-2 TV Interface Board**	**349.00**

**We do honor Synertek discount coupons*

QUALITY EXPANSION BOARDS DESIGNED SPECIFICALLY FOR KIM-1, SYM-1 & AIM 65

These boards are set up for use with a regulated power supply such as the one below, but, provisions have been made so that you can add onboard regulators for use with an unregulated power supply. But, because of unreliability, we do not recommend the use of onboard regulators. All I.C.'s are socketed for ease of maintenance. *All boards carry full 90-day warranty.*

All products that we manufacture are designed to meet or exceed industrial standards. All components are first qualtiy and meet full manufacturer's specifications. All this and an extended burn-in is done to reduce the normal percentage of field failures by up to 75%. To you, this means the chance of inconvenience and lost time due to a failure is very rare; but, if it should happen, we guarantee a turn-around time of less than forty-eight hours for repair.

Our money back guarantee: If, for any reason you wish to return any board that you have purchased directly from us within ten (10) days after receipt, complete, in original condition, and in original shipping carton; we will give you a complete credit or refund less a $10.00 restocking charge per board.

VAK-1 8-SLOT MOTHERBOARD

This motherboard uses the KIM-4* bus structure. It provides eight (8) expansion board sockets with rigid card cage. Separate jacks for audio cassette, TTY and power supply are provided. Fully buffered bus.

VAK-1 Motherboard **$129.00**

VAK-2/4 16K STATIC RAM BOARD

This board using 2114 RAMs is configured in two (2) separately addressable 8K blocks with individual write-protect switches.

VAK-2 16K RAM Board with only 8K of RAM (½ populated)	**$239.00**
VAK-3 Complete set of chips to expand above board to 16K	**$175.00**
VAK-4 Fully populated 16K RAM	**$379.00**

VAK-5 2708 EPROM PROGRAMMER

This board requires a +5 VDC and ±12 VDC, but has a DC to DC multiplyer so there is no need for an additional power supply. All software is resident in on-board ROM, and has a zero-insertion socket.

VAK-5 2708 EPROM Programmer **$269.00**

VAK-6 EPROM BOARD

This board will hold 8K of 2708 or 2758, or 16K of 2716 or 2516 EPROMs. *EPROMs not included.*

VAK-6 EPROM Board **$129.00**

VAK-7 COMPLETE FLOPPY-DISK SYSTEM (May '79)

VAK-8 PROTYPING BOARD

This board allows you to create your own interfaces to plug into the motherboard. Etched circuitry is provided for regulators, address and data bus drivers; with a large area for either wire-wrapped or soldered IC circuitry.

VAK-8 Protyping Board **$49.00**

POWER SUPPLIES

ALL POWER SUPPLIES are totally enclosed with grounded enclosures for safety, AC power cord, and carry a full 2-year warranty.

FULL SYSTEM POWER SUPPLY

This power supply will handle a microcomputer and up to 65K of our VAK-4 RAM. ADDITIONAL FEATURES ARE: Over voltage Protection on 5 volts, fused, AC on/off switch. Equivalent to units selling for $225.00 or more.

Provides +5 VDC @ 10 Amps & ±12 VDC @ 1 Amp
VAK-EPS Power Supply **$125.00**

KIM-1* Custom P.S. provides 5 VDC @ 1.2 Amps and +12 VDC @ .1 Amps
KCP-1 Power Supply **$41.50**

SYM-1 Custom P.S. provides 5 VDC @ 1.4 Amps
VCP-1 Power Supply **$41.50**

*KIM is a product of MOS Technology

RNB ENTERPRISES INCORPORATED

2967 W. Fairmount Avenue
Phoenix AZ 85017
(602)265-7564

SUPER HI-LO FOR THE SYM-1

Jack Gieryic
2041 138th Ave. N.W.
Andover, MN 55303

Super HI-Lo has a new twist to the game. This program fits into the standard 1K SYM and execution begins at location 200. The left two LED digits are your upper limit (initialized to 99) and the middle two digits are your lower limit (initialized to 00). SYM picks a random number and you attempt to guess it. Your attempt count is seen in the right two digits. The right digit will blink when it's your last guess.

After entering the command GO 200 CR press any key to start the contest. Enter your two digit guess (decimal only) and hit the "A" key. Win or loose you get an appropriate message at the end after which the LED's go blank. Hit any key and you are ready for a second game. If you didn't guess the number then you will be given one more chance in the next game. If you are lucky enough to guess the number then you will have one less chance the next game.

For you SYMMERS who are interested in taking things one step further, you will find MESSAG an interesting subroutine you may want to incorporate in your own programs. This code is entirely relocatable except for the first four instructions which must be calculated if the code is moved. The routine uses page zero locations OD, OE, OF and 10, but you can change that too if necessary. The A and X registers contain the message buffer address per comments in the program. This message buffer contains segment codes which will light up any combination of LED segments.

Refer to Figure 4-6 Keyboard/Display Schematic in your reference manual for the LED segments in the lower right corner. Segment "a" is turned on by setting bit 0 to a one in a message buffer entry. Segment "b" is controlled by bit 1 and so on with segments c, d, e, f, g and the decimal point. Thus a hex 5C is a lower case O (segments c, d, e, and g). Feel free to change either message but don't forget to add a few OO characters at the start and end of your message. If you relocate the message buffer then change the register parameters prior to the call to MESSAG.

One other note on the program. By changing the value at location 206 you can alter the rate at which the right LED will blink when you reach your last chance.

```
                    SYM SUPER HI-LO
                    JOHN GIERYIC
                    APRIL 1979

                    SYM REFERENCES

035E                KYSTAT *      $896A
035E                ACCESS *      $8B86
035E                OUTBYT *      $82FA
035E                SCAND  *      $8906
035E                KEYQ   *      $8923
035E                GETKEY *      $88AF
035E                ASCNIB *      $8275
035E                DISBUF *      $A640
035E                RDIG   *      $A645

                    MESSAGE POINTERS

035E                MFAIL  *      $0360
035E                MSUCC  *      $0380

0000                       ORG    $0000

0000 00             UPP    =      $00    UPPER NUMBER
0001 00             LOW    =      $00    LOWER NUMBER
0002 00             ACNT   =      $00    ATTEMPT COUNT
0003 00             RAN    =      $00    RANDOM NUMBER 2 - 98
0004 00             TEMP   =      $00
0005 00             UGES   =      $00    GUESS UNITS
0006 00             TGES   =      $00    GUESS TENS
0007 00             BLINK  =      $00    BLINK FLAG 1 = BLINK
0008 00             TDIG   =      $00    SAVE RDIG
0009 00             DARK   =      $00    1 = DARK
000A 00             LATT   =      $00    ATTEMPT LIMIT
000B 00             ONOFF  =      $00    BLINKING
000C 00             BLIM   =      $00    BLINKING LOOP COUNT INIT.
000D 00             COUNT  =      $00
000E 00             LOOPA  =      $00
000F 00             LOOPB  =      $00
0010 00             CLIM   =      $00    MESSAGE LIMIT

0200                       ORG    $0200  PROGRAM ORIGIN

0200 20 86 8B       BEGIN  JSR    ACCESS
0203 A9 60                 LDAIM  $60    INIT BLINKING LOOP LIMIT
0205 85 0C                 STA    BLIM
0207 A9 06                 LDAIM  $06    INIT ATTEMPT COUNTER
0209 85 0A                 STA    LATT

020B A9 63          TILL   LDAIM  $63    INIT UPPER LIMIT
020D 85 00                 STA    UPP
020F A9 00                 LDAIM  $00    INIT BLINK FLAG
0211 85 07                 STA    BLINK
0213 85 01                 STA    LOW    LOWER LIMIT
0215 85 02                 STA    ACNT   ATTEMPT COUNT
0217 A9 01                 LDAIM  $01
```

```
0219 85 03          STA    RAN    RANDOM NUMBER

021B E6 03    INCRAN INC    RAN    INCREMENT RANDOM NUMBER
021D A5 03          LDA    RAN
021F C9 63          CMPIM $63    IF EQUAL 99 DECIMAL
0221 D0 04          BNE    KEYIN
0223 A9 02          LDAIM $02    THEN RESET TO 2
0225 85 03          STA    RAN

0227 20 6A 89 KEYIN  JSR    KYSTAT IS A KEY DOWN?
022A 90 EF          BCC    INCRAN LOOP UNTIL ONE IS DOWN
022C A5 00    LIMITS LDA    UPP    PUT UPPER, LOWER AND
022E 20 00 03       JSR    HTDEC  ATTEMPT COUNT IN
0231 20 FA 82       JSR    OUTBYT DISPLAY BUFFER
0234 A5 01          LDA    LOW
0236 20 00 03       JSR    HTDEC
0239 20 FA 82       JSR    OUTBYT
023C A5 02          LDA    ACNT
023E 20 00 03       JSR    HTDEC
0241 20 FA 82       JSR    OUTBYT

0244 20 06 89 DISP   JSR    SCAND  LIGHT LED
0247 20 23 89       JSR    KEYQ   IF KEY IS DOWN,
024A D0 30          BNE    READK
024C A5 07          LDA    BLINK  IF BLINKING IS REQUESTED
024E C9 01          CMPIM $01
0250 D0 F2          BNE    DISP
0252 A5 0B          LDA    ONOFF  IF TIME TO TURN CHARACTER ON
0254 D0 21          BNE    INCLOP
0256 A5 09          LDA    DARK   IF TURN CHAR. OFF
0258 C9 01          CMPIM $01
025A D0 0E          BNE    RIGHT
025C AD 45 A6       LDA    RDIG   THEN GET CHARACTER
025F 85 08          STA    TDIG   SAVE IT
0261 A9 00          LDAIM $00    SET RIGHT DIGIT BLANK
0263 8D 45 A6       STA    RDIG
0266 C6 09          DEC    DARK   SWITCH FLAG
0268 F0 07          BEQ    LCOUNT
026A A5 08    RIGHT  LDA    TDIG   ELSE RESTORE RIGHT DIGIT
026C 8D 45 A6       STA    RDIG
026F E6 09          INC    DARK   SWITCH FLAG
0271 A5 0C    LCOUNT LDA    BLIM   RESET LOOP COUNTER
0273 85 0B          STA    ONOFF
0275 D0 CD          BNE    DISP
0277 E6 0B    INCLOP INC    ONOFF  INCR. LOOP COUNTER
0279 4C 44 02       JMP    DISP   LOOP

027C 20 AF 88 READK  JSR    GETKEY GET DEPRESSED KEY
027F 20 75 82       JSR    ASCNIB
0282 C9 0A          CMPIM $0A    IS IT "A" (ATTEMPT)
0284 F0 0B          BEQ    SETLOP YES
0286 AA             TAX           NO
0287 A5 05          LDA    UGES   MOVE PREVIOUS KEY
0289 85 06          STA    TGES   TO TENS DIGIT
028B 8A             TXA
028C 85 05          STA    UGES   PUT NEW KEY INTO UNITS
```

```
028E 4C 44 02          JMP    DISP   LOOP

0291 A6 06      SETLOP LDX    TGES   SET LOOP INDEX (TENS)
0293 A9 00             LDAIM $00     INIT A REGISTER
0295 18                CLC           CLEAR CARRY FALG
0296 CA         DECX   DEX           DECR. X REG.
0297 30 04             BMI    ADUNIT IF NEG, THEN FINISHED

0299 69 0A             ADCIM $0A     ELSE ADD 10
029B D0 F9             BNE    DECX    LOOP
029D 65 05      ADUNIT ADC    UGES   ADD UNITS VALUE
029F C5 03             CMP    RAN    COMPARE TO RANDOM
02A1 D0 03             BNE    ADUP
02A3 4C E4 02          JMP    SUCEED GUESS = RANDOM

02A6 90 09      ADUP   BCC    TLOW
02A8 C5 00             CMP    UPP
02AA B0 0B             BCS    INCA
02AC 85 00      RUP    STA    UPP    REPLACE UPPER WITH GUESS
02AE 4C B7 02          JMP    INCA
02B1 C5 01      TLOW   CMP    LOW
02B3 90 02             BCC    INCA
02B5 85 01             STA    LOW    REPLACE LOWER WITH GUESS
02B7 E6 02      INCA   INC    ACNT   INCR. ATTEMPT COUNT
02B9 A5 02             LDA    ACNT   LIMIT REACHED?
02BB C5 0A             CMP    LATT
02BD D0 03             BNE    TEST   NO
02BF 4C D8 02          JMP    FAIL   YES = FAILURE
02C2 38         TEST   SEC
02C3 A5 0A             LDA    LATT   LAST ATTEMPT COMING UP
02C5 E5 02             SBC    ACNT
02C7 C9 01             CMPIM $01
02C9 D0 0A             BNE    WAIT   NO
02CB E6 07             INC    BLINK  YES - INIT FOR BLINKING
02CD A5 0C             LDA    BLIM
02CF 85 0B             STA    ONOFF
02D1 A9 01             LDAIM $01
02D3 85 09             STA    DARK
02D5 4C 2C 02   WAIT   JMP    LIMITS GO WAIT FOR NEXT ATTEMPT

02D8 E6 0A      FAIL   INC    LATT   FAILURE = INCR ATTEMPT LIMIT
02DA A2 03             LDXIM MFAIL   / MESSAGE HI BYTE
02DC A9 60             LDAIM MFAIL    MESSAGE LO BYTE
02DE 20 17 03          JSR    MESSAG DISPLAY FAILURE MESSAGE
02E1 4C 0B 02          JMP    TILL   RESTART HI-LO

02E4 C6 0A      SUCEED DEC    LATT   SUCCESS = DECR ATTEMPT LIMIT
02E6 A2 03             LDXIM MSUCC   / MESSAGE HI BYTE
02E8 A9 80             LDAIM MSUCC    MESSAGE LO BYTE
02EA 20 17 03          JSR    MESSAG DISPLAY SUCCESS MESSAGE
02ED 4C 0B 02          JMP    TILL   RESTART HI-LO

                SUBROUTINE HTDEC

                ENTRY  JSR HTDEC
```

```
                        THIS ROUTINE WILL CONVERT A HEX NUMBER
                        TO DECIMAL.  UPON ENTRY THE A REGISTER CONTAINS
                        THE NUMBER TO CONVERT.  UPON EXIT THE A REG.
                        CONTAINS THE UNITS DIGIT AND THE X REGISTER
                        CONTAINS THE TENS DIGIT.

0300                           ORG   $0300

0300 A2 00          HTDEC  LDXIM $00      INIT TENS COUNT
0302 38                    SEC
0303 E9 0A          HTA    SBCIM $0A      SUBTRACT 10 DECIMAL
0305 30 03                 BMI   HTB
0307 E8                    INX            INCR. TENS DIGIT
0308 D0 F9                 BNE   HTA
030A 69 0A          HTB    ADCIM $0A      UNITS DIGIT
030C 85 04                 STA   TEMP
030E 8A                    TXA
030F 18                    CLC
0310 2A                    ROLA
0311 2A                    ROLA
0312 2A                    ROLA
0313 2A                    ROLA
0314 65 04                 ADC   TEMP
0316 60                    RTS

                        SUBROUTINE  MESSAG

                        ENTRY  JSR MESSAG

                        THIS ROUTINE WILL PARADE THE MESSAGE SPECIFIED
                        BY THE CALLER ACROSS THE LEDS.  THE A REGISTER
                        CONTAINS THE LO BYTE OF THE MESSAGE ADDRESS.  THE
                        X REG. CONTAINS THE HI BYTE OF THE MESSAGE ADDRESS.
                        THE FIRST BYTE OF THE MESSAGE CONTAINS THE NUMBER
                        OF BYTES IN THE MESSAGE MINUS 5.  THIS COUNT
                        INCLUDES THE FIRST BYTE

0317 8D 24 03   MESSAG STA   MAD    +01 CHANGE INSTRUCTION
031A 8E 25 03          STX   MAD    +02
031D 8D 37 03          STA   MADX   +01 CHANGE INSTRUCTION
0320 8E 38 03          STX   MADX   +02
0323 AD FF FF   MAD    LDA   $FFFF  ADDRESS WILL BE CHANGED
0326 85 10             STA   CLIM
0328 A9 00             LDAIM $00
032A 85 0D             STA   COUNT
032C 85 0E             STA   LOOPA
032E 85 0F             STA   LOOPB
0330 E6 0D             INC   COUNT
0332 A4 0D      MESS   LDY   COUNT
0334 A2 00             LDXIM $00
0336 B9 FF FF   MADX   LDAY  $FFFF  ADDRESS WILL BE CHANGED
0339 9D 40 A6          STAX  DISBUF
033C C8                INY
033D E8                INX
033E E0 06             CPXIM $06
0340 D0 F4             BNE   MADX
```

```
0342 E6 0D              INC    COUNT
0344 20 06 89  MESSA    JSR    SCAND
0347 E6 0E              INC    LOOPA
0349 D0 F9              BNE    MESSA
034B E6 0F              INC    LOOPB
034D A5 0F              LDA    LOOPB
034F C9 02              CMPIM  $02
0351 D0 F1              BNE    MESSA
0353 A5 0E              LDA    LOOPA
0355 85 0F              STA    LOOPB
0357 A5 0D              LDA    COUNT
0359 C5 10              CMP    CLIM
035B D0 D5              BNE    MESS
035D 60                 RTS
```

THE FAILURE MESSAGE BEGINS AT LOCATION 0360. THE FIRST BYTE IS THE HEX NUMBER OF BYTES IN THE MESSAGE MINUS FIVE. THE MESSAGE IS IN THE FORM OF SEGMENT CODES. A MEMORY LISTING FOLLOWS. LOAD THIS BEGINNING AT LOCATION 0360.

```
0360  0B 00 00 6E 3F 3E 00 38 3F 3F
0368  3F 3F 6D 79 00 00 00 00
```

THE SUCCESS MESSAGE BEGINS AT LOCATION 0380.

```
0380  08 00 00 39 5C 50 50 79
0388  58 78 00 00 00
```

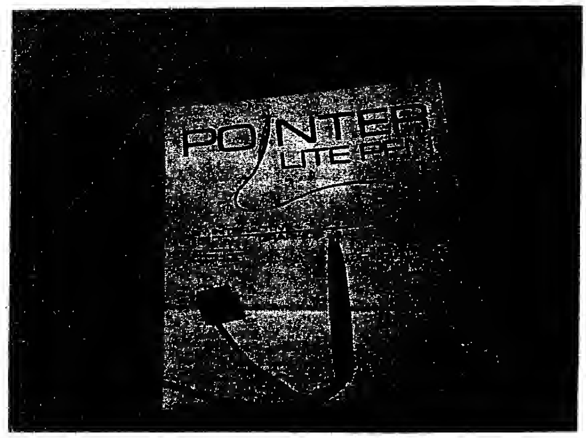
Computer Forum

· BUSINESS
· EDUCATIONAL
· PERSONAL

14052 EAST FIRESTONE BOULEVARD · SANTA FE SPRINGS, CALIFORNIA 90670

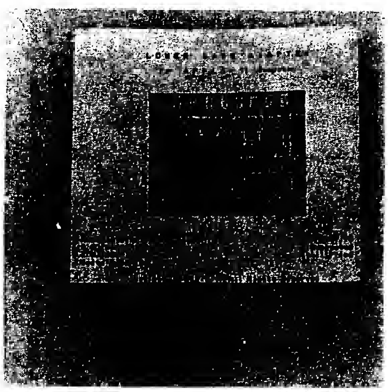
(213) 921-2111 — (714) 739-0711

LIGHT PEN FOR THE APPLE—#035-03610 .... **$ 34.95**



Plugs into the game paddle connector. It includes three demo programs:

1. This demonstration program is a tutorial on the use of the lite pen as a menu selection tool. It is self prompting and instructing and is a perfect example of a realistic lite pen application.
2. This demonstration program is a "low-resolution" graphics demonstration which allows the user to select from a menu of "high-resolution" shapes. In addition, the user is also given the capability of selecting colors from a color menu. Selection from either of the two menus is accomplished by depressing the RETURN key. To place the selected shape on the screen, depress the RETURN key.
3. This demonstration program is a "low-resolution" graphics color bit-pad demonstration. A color menu is displayed and user selects a color by depressing any key. To place the color on the screen, depress any key. To clear the working screen the user depresses the ESC key.



LOWER CASE ADAPTER FOR THE APPLE II, #042-04479 .... **$ 49.95**
WORD PROCESSOR Program that uses the lower case adaptor, #042-04420 .... **$ 59.95**
The lower case adapter will only work in the normal field mode. Inverse or flashing lower case characters will not work. This adapter will increase your Apple into a 96 character character-set. It also includes a few other characters like { ¦ } ~ ■ ,

COMMODORE 8K PETS, 8 LEFT AT THE REDUCED PRICE OF ...... **$ 699.95**
Plus $ 15.00 for Shipping & Handling
Call For Confirmation and Reservation

16K RAM FOR APPLE II, 200NS or Better .... **$ 75.00**
Plus $ 2.00 for Shipping & Handling

PROGRAM DESIGN INC. Software is probably the best written educational software available. We have the complete line of P.D.I. software for the PET™ and the APPLE II. Following is a list of titles available from P.D.I.:

VOCABULARY BLDR. I, #016-01259, 16K INT BASIC: First tape in a two tape course. 10 lessons & 1 quiz ea. tape .... **$ 13.50**
VOCABULARY BLDR. II, #016-01314, 16K INT BASIC: The second tape in a two tape course. 10 lessons & 1 quiz on ea. tape .. **$ 13.50**
PRESCHOOL IQ-BUILDER, #016-01195, 16K INT BASIC: 6 lessons: "Same & Different" & 1 program letter builder .... **$ 13.50**
STEP BY STEP, #016-01356, 16K APPLESOFT II: A tutorial program in computer language of Applesoft II basic. 10 lessons ... **$ 39.95**
WORD MASTER/STORY BUILDER, #016-01399, 16K INT BASIC: Thinking person's game. Nouns, verbs & adjectives .... **$ 13.50**
MEMORY BUILDER, #016-01410, 16K INT BASIC: A concentration game with 20 boxes on screen .... **$ 13.50**

Following is a list of software companies that we deal with. (We are adding new software every month).

APPLE Software Bank • Speakeasy • Powersoft • Forum • Southeastern Software • Wise Owl Workshop • Program Design Inc. Programma International • Personal Software • Quality Software • Peripherals Unlimited Software • George W. Lee Software • Pet Shack Creative Computing • ZZYP-PAX • Softape

Computer Forum
• BUSINESS
• EDUCATIONAL
• PERSONAL
14052 East Firestone Blvd. • Santa Fe Springs, CA 90670
(213) 921-2111 • (714) 739-0711

NAME
ADDRESS
CITY STATE
ZIP PHONE

STOCK NO. & DESCRIPTION	PRICE	QUANTITY	AMOUNT

Yes, you may charge it!
B of A Card Mastercharge

Expiration Date ____________
Card # ____________
Signature ____________

Subtotal ______
Calif. residents 6% tax ______
Shipping ______
TOTAL AMOUNT ENCLOSED ______

A 100 uS 16 CHANNEL ANALOG TO DIGITAL CONVERTER FOR 65XX MICROCOMPUTER SYSTEMS

J. C. Williams
55 Holcomb St.
Simsbury, CT 06070

Analog to digital (A/D) conversion can be useful in many microcomputer systems. The design presented here takes advantage of a large scale integrated circuit, the ADC0817, to simplify a 16 channel, 8 bit A/D system which can be attached to the bus of 65XX microcomputers. The applications that I have found for this system have included "straight" data acquisition, game joystick position reading, graphic input generation and voice recognition. Of course, the software for each of these applications is different, but they all require multichannel, reasonably fast A/D. The 100 us conversion time of this system depends only on the 1 MHz clock frequency of the microcomputer. The microprocessor is not involved in the A/D conversions. Once the conversion is started, the processor can work on other tasks until the digital result is available.

The Hardware

This device appears to the programmer as a block of memory starting at a base address, BASE, and extending through 16 locations to BASE + 15. (The actual circuit described occupies 256 locations because of incomplete decoding.) An analog to digital conversion of a selcted channel, say channel X, is started by writing to BASE + X. The 8 bit conversion result may then be read from any location in the block (eg. BASE) any time after the 100 US conversion time has elapsed. If desired, the end of conversion signal from the ADC0817 may cause an interrupt to get the attention of the processor. If multiple A/D conversions at the maximum speed are required the 65XX can be kept busy with "housekeeping" during the conversion delay time. The example programs illustrate two ways the converter may be driven. The system uses just five integrated circuits and can be built for less than $40. The design, shown in Figure 1, occupies a six square inch area on a Vector plugboard and draws only 60 mA of current from the +8 Volt DC unregulated power supply. Operation of the circuit is simple because the ADC0817 performs all analog switching and A/D functions. The base address of the converter is fixed by six switches attached to the DM8131 six bit comparitor.

When the processor accesses memory locations having address bits A15-A10 matching the switch settings, the DM8131 output goes low. This output is NOR'ed with A9 and A8 to further reduce the memory space occupied by the circuit to one 65XX page. The possible base addresses which can be obtained with this decoder can fall on any 1K boundary and A9 and A8 must be "O's". For example, base addresses (in hex) can be set to A000 or A400 but not A100, A200, or A300. In the design drawn, A9 and A8 must be low for the A/D to be selected, but this could be changed if A9 and/or A8 were inverted using unused sections of the 74LS05.

When the A/D is selected, the output of the NOR gate (pin 12 of the 74LS27) goes to a "1"; this can be used as a "board selected" signal if needed (eg. by KIM-1 users for DECODE ENABLE). The microprocessor R/W and 02 lines, along with an inverted board select signal and combined in two NOR gates which 1) latch channel select bits A3-A0 and start A/D conversion during 02 of write cycles and 2) enable the tri-state data bus drivers during 02 of read cycles. The end of conversion (EOC) signal, produced by the ADC0817 when the most recent conversion has been completed, can be connected to a processor interrupt line through one of the 74LS05 open collector inverters. These interrupts must be cleared by starting another A/D conversion.

Wire-wrap construction is suitable for the circuit and component layout is not critical. It is good practice, however, to orient the analog input area away from digital circuits. The REF + and REF- reference voltages must not be noisy if the full accuracy, 20 mV per bit, is to be achieved. The +5 Volt regulator should not be shared with other circuitry. The layout used in one of the prototypes is sketched in Figure 2. Figure 2 also shows several input connections which may be useful. The circuit has two limitations: 1) input voltages must be between 0 and +5 Volts and 2) signals being converted should not change appreciably during the 100 us conversion period. Both of these limitations may be eliminated by appropriate analog conditioning circuitry, but the simplicity of the design is lost. Builders who want to add features to the circuit should consult the ADC0817 specification and application information.

The Software

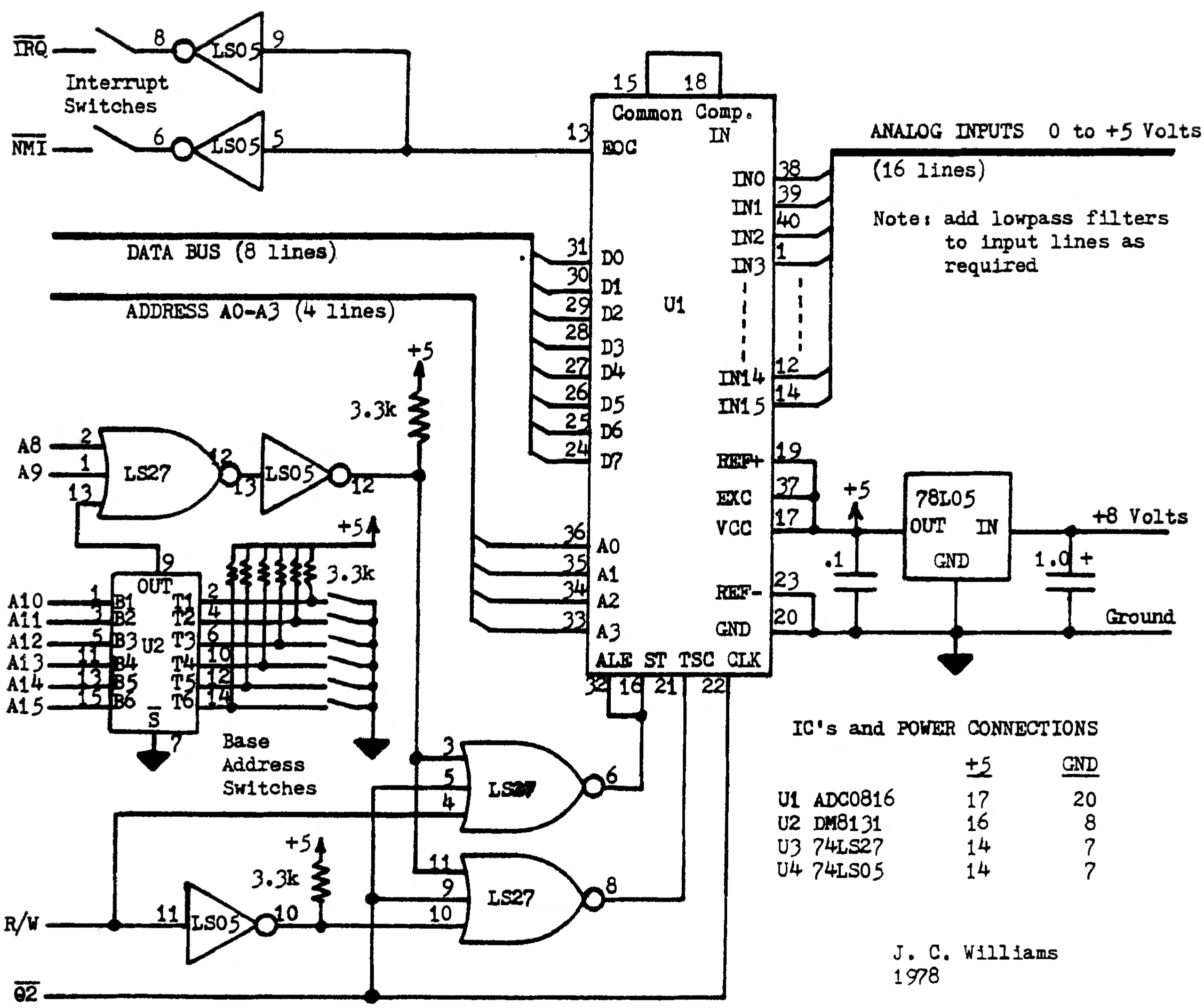
Two example subroutines which use the A/D converter illustrate how it is handled by software. The program which calls the A/D subroutine must initialize both the channel selection and storage defining parameters before the JSR instruction is executed. In the examples, an index register contains the channel selection information because of the ease of using an indexed addressing mode to start a conversion. Data storage is either on page 0 or pointed to by page 0 variables. The A/D subroutines must either contain delays or take enough time between writing to and reading from the ADC0817 to allow it to finish the conversion. Components for this very useful piece of hardware can be obtained from a number of sources readily available to low-volume users. Both National Semiconductor and Texas Instruments produce the ADC0817 and its more accurate counterpart, the ADC0816. The ADC0817 and its data sheet have been recently listed by TRI-TEK, Inc., 7808 N. 27th Ave., Phoenix, AZ 85021. Many other suppliers, such as Jameco Electronics, 1021 Howard Avenue., San Carlos, CA 94979, and Advanced Computer Products, 1310 "B" E. Edinger, Santa Ana, CA 92713, can supply the other components.

```
                      MCAD - MULTI-CHANNEL A/D CONVERSION
                      J. C. WILLIAMS
                      JANUARY 1979

0200                         ORG   $0200

0200              BASE   *     $B000  BASE ADDRESS OF ADC0816
0200              STORE  *     $9000  START OF 16 BYTE STORAGE AREA

0200 9D 00 B0     MCAD   STAX  BASE   START CONVERSION ON CHANNEL X
0203 A0 0E               LDYIM $0E    DELAY FOR CONVERSION,
0205 88           DY     DEY          MINIMUM VALUE = $0E
0206 D0 FD               BNE   DY
0208 AD 00 B0            LDA   BASE   GET CONVERTED DATA
020B 9D 00 90            STAX  STORE  STORE DATA
020E CA                  DEX
020F 10 EF               BPL   MCAD   DO NEXT CHANNEL
0211 60                  RTS          FINISHED

                  EXAMPLE CALLING ROUTINE FOR MCAD

0212 A2 0F        MCMAIN LDXIM $0F    SELECT CONVERSION OF ALL
0214 20 00 02            JSR   MCAD   16 CHANNELS AND GO TO SUBROUTINE
0217 00                  BRK          EXIT ** BE SURE TO INIT IRQ VECTOR **
```

```
                     CXAD SUBROUTINE
                     J. C. WILLIAMS
                     JANUARY 1979

0300                          ORG    $0300

0300                 BASE     *      $B000  BASE ADDRESS OF ADC0816
0300                 SP       *      $0000  STORAGE POINTER
0300                 SPSTR    *      $0002  LOC OF STORAGE BLOCK START ADDRESS
0300                 SPSTP    *      $0004  LOC OF STORAGE BLOCK END ADDRESS

0300 9D 00 B0        CXAD     STAX   BASE   START FIRST CONVERSION
0303 A5 02                    LDAZ   SPSTR  INIT STORAGE POINTER
0305 85 00                    STAZ   SP
0307 A5 03                    LDAZ   SPSTR  +01
0309 85 01                    STAZ   SP     +01
030B D8                       CLD           USE BINARY MODE
030C A0 05                    LDYIM  $05    INSERT DELAY TO ALLOW
030E 88              DY       DEY           INITIAL CONV. TO COMPLETE
030F D0 FD                    BNE    DY
0311 F0 16                    BEQ    DELAY
0313 A5 00           TSTEND   LDAZ   SP     TEST FOR END OF
0315 C5 04                    CMPZ   SPSTP  STORAGE BLOCK
0317 A5 01                    LDAZ   SP     +01
0319 E5 05                    SBCZ   SPSTP  +01
031B B0 1D                    BCS    RT
031D A9 01                    LDAIM  $01    ADD ONE TO STORAGE POINTER
031F 65 00                    ADCZ   SP
0321 85 00                    STAZ   SP
0323 A9 00                    LDAIM  $00
0325 65 01                    ADCZ   SP     +01
0327 85 01                    STAZ   SP     +01
0329 A0 05           DELAY    LDYIM  $05    DELAY TO FIX TIME BETWEEN CONV'S.
032B 88              DYA      DEY
032C D0 FD                    BNE    DYA
032E AD 00 B0                 LDA    BASE   READ CONVERTED RESULT
0331 9D 00 B0                 STAX   BASE   START NEXT CONVERSION IMMEDIATELY
0334 A0 00                    LDYIM  $00    SET STORAGE OFFSET
0336 91 00                    STAIY  SP     STORE RESULTS
0338 F0 D9                    BEQ    TSTEND ALWAYS TAKEN
033A 60              RT       RTS

                     EXAMPLE CALLING ROUTINE FOR CXAD

033B A2 00           CXMAIN   LDXIM  $00    SELECT CHANNEL 0
033D A9 00                    LDAIM  $00    SET STARTING ADDRESS OF
033F 85 02                    STAZ   SPSTR  STORAGE BLOCK TO $9000
0341 A9 90                    LDAIM  $90
0343 85 03                    STAZ   SPSTR  +01
0345 A9 FF                    LDAIM  $FF    SET ENDING ADDRESS OF
0347 85 04                    STAZ   SPSTP  STORAGE BLOCK TO $9FFF
0349 A9 9F                    LDAIM  $9F
034B 85 05                    STAZ   SPSTP  +01
034D 20 00 03                 JSR    CXAD
0350 00                       BRK           EXIT   ** BE SURE TO INIT IRQ VECTOR **
```

FIGURE 1

16 CHANNEL ANALOG TO DIGITAL CONVERTER SYSTEM
FOR 65XX MICROPROCESSOR SYSTEMS



IC's and POWER CONNECTIONS

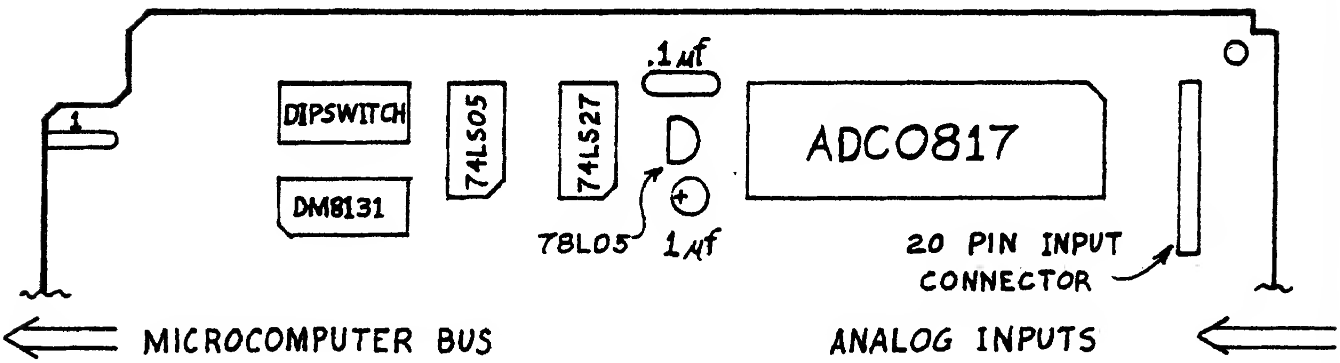
		+5	GND
U1	ADC0816	17	20
U2	DM8131	16	8
U3	74LS27	14	7
U4	74LS05	14	7

J. C. Williams
1978

FIGURE 2

16 CHANNEL A/D CONVERTER FOR 65XX SYSTEMS

COMPONENT SIDE OF 6.5" X 4.5" PROTOTYPING CARD - VECTOR 3662

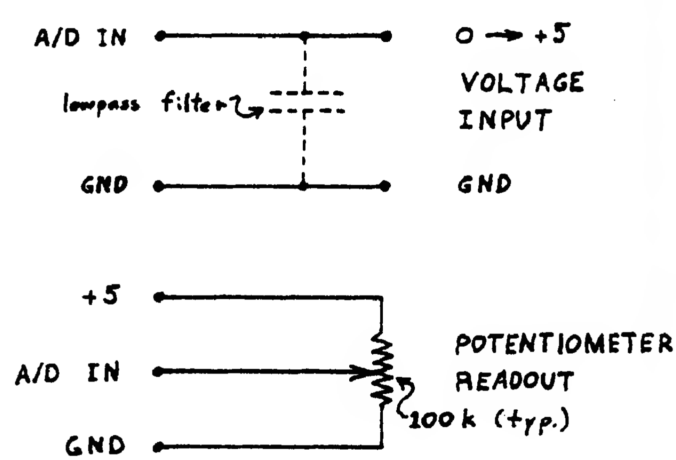


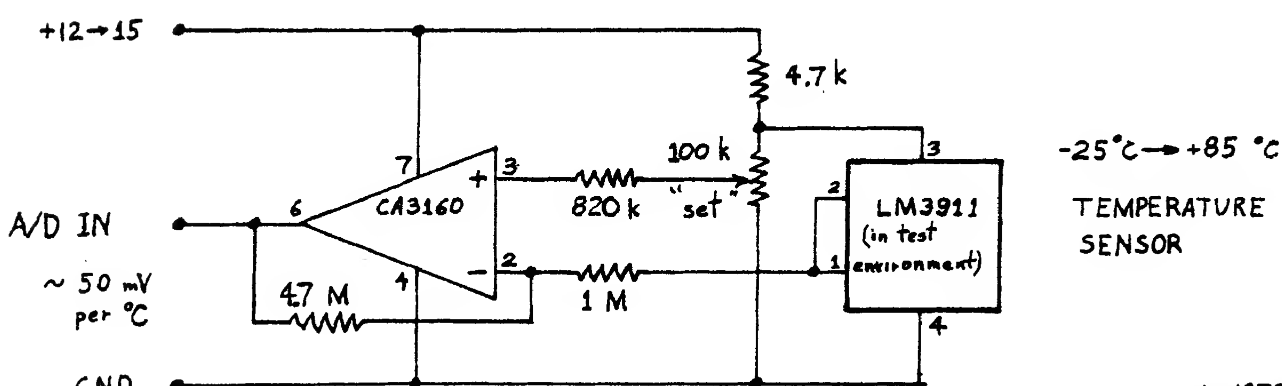
INPUT CONNECTOR DETAIL

TOP VIEW

GND	+5
IN14	IN15
IN12	IN13
IN10	IN11
IN8	IN9
IN6	IN7
IN4	IN5
IN2	IN3
IN0	IN1
GND	+5

APPLICATIONS





JCW - 1979

REAL-TIME GAMES ON OSI

David Morganstein
9523 48th Place
College Park, MD 20704

This note discusses how real-time games can be written for OSI Challenger systems which use a serial terminal run from the ACIA. The terminal in my system is an ADM-3A, but the same principal applies to any other. The sample program which is included does use the cursor control procedure of the ADM-3A, but it is a common enough terminal that many readers will be able to use it directly. The cursor control is accomplished in a one-line subroutine and can be changed to another procedure easily.

My original goal was to write video games, but I did not have a separate TV monitor, 440 video board and A/D convertor to do this. Fortunately, there was a way!! First, I'll discuss a procedure for polling the serial terminal keyboard and then the video display on the terminal.

The basic idea was to use a PEEK command rather than an IMPUT statement. That way the program does not have to stop while the player ponders his response. This was the ONLY way to play Lunar Lander. The typical version gives the Captain unlimited time to ponder his response and minimizes crash landings. Several articles in BYTE and elsewhere talk about using A/D convertors and joysticks. Of course, this is a fine way to go, but the same effect can be created without the added hardware.

The input byte from the ACIA appears at $FC01. To get a little appreciation for this, look at the ROM monitor routine starting at $FE00, this is called INCH in the OSD documentation. (See Figure 1.) By peeking at 64513 ($FC01), you can read the byte sent by the terminal. The only problem with this is the parity bit. That is, the bytes indicating the numbers 0-9 do not increase smoothly but have bit 7 set or not to insure parity. You can solve this by subtracting 128 when the PEEK (64513) is greater than 128. In the INCH routine this is accomplished with an AND #$7F, masking bit 7. In this way, you get values from 48 to 57 for the keys 0-9. Now these values can be used to change the burn rate of the lunar lander.

The program is fairly short and is generally self-explanatory. The polling is done in subroutine 5000. The test for 13 is needed since this is a null byte appearing before any keyboard entry has been made. As it now runs, extra boost can be given by typing a non-numeric. This should probably be prevented since it will allow a "sinking ship" to be saved, most unsporting!!

The other interesting feature is the cursor control. This is accomplished in line 6000. The ADM-3A requires two control bytes be sent, CHR$(27) and CHR$(61), in order to set up the X and Y coordinates which follow. As given in the subroutine, the X value can be from 1 to 80 and the Y from 1 to 24, which correspond to the column and row (counting from the top left) of the position to be printed. Be careful when using this to not exceed these ranges. The cursor control is used to set-up a "lander control panel" and then update the "meter readings" as the play progresses.

If your wondering what line 500 does, its used for timing. By adjusting the variable DE(lay), the speed of the game can be changed slightly. I was shooting for a twice per second update on the panel. Unfortunately, when the LOW FUEL WARNING comes on the timing changes. Well, you can't have everything. (I'm sure somebody out there will figure out how to correct this....)

```
FE00                    ORG    $FE00

FE00 AD 00 FC   START   LDA    $FC00
FE03 4A                 LSRA
FE04 90 FA              BCC    START
FE06 AD 01 FC           LDA    $FC01
FE09 29 7F              ANDIM  $7F
FE0B 48                 PHA
FE0C AD 00 FC           LDA    $FC00
FE0F 4A                 LSRA
FE10 4A                 LSRA
FE11 90 F9              BCC    $FE0C
FE13 68                 PLA
FE14 8D 01 FC           STA    $FC01
FE17 60                 RTS
FE18 20 00 FE           JSR    START
FE1B C9 52              CMPIM  $52
FE1D F0 16              BEQ    $FE35
FE1F C9 30              CMPIM  $30
FE21 30 F5              BMI    $FE18
FE23 C9 3A              CMPIM  $3A
FE25 30 0B              BMI    $FE32
FE27 C9 41              CMPIM  $41
FE29 30 ED              BMI    $FE18
FE2B C9 47              CMPIM  $47
```

```
100 PRINTCHR$(26):X=25:Y=10:GOSUB6000
104 PRINT"L U N A R   L A N D E R ":Y=12:GOSUB6000
106 INPUT"DO YOU NEED INSTRUCTIONS (Y/N) ";N$
110 IFN$="N"GOTO190
115 PRINT:PRINT
120 PRINTTAB(10)"THIS IS A REAL TIME LUNAR LANDER SIMULATION.
130 PRINTTAB(10)"TO PLAY, MERELY ENTER THE POUNDS OF
140 PRINTTAB(10)"FUEL WHICH YOU WISH TO BURN BY TYPING A DIGIT (0-9).
150 PRINTTAB(10)"THE NINE GIVES MAXIMUM BURN, SLOWING YOU DOWN AT THE
155 PRINTTAB(10)"FASTEST RATE.  A ZERO GIVES NO BURN AND LETS YOU FRE
160 PRINTTAB(10)"FALL.":PRINT:INPUT"   READY...TYPE GO  ";N$
190 PRINTCHR$(26):Y=4:X=28:GOSUB6000:PRINT"TIME TO FUEL EXHAUSTION"
200 X=20:Y=7:GOSUB6000:PRINT"BURN RATE"
220 X=50:GOSUB6000:PRINT"FUEL"
230 Y=8:X=20:GOSUB6000:PRINT(LBS/SEC)"X=50:GOSUB6000:PRINT"(LBS)"
240 Y=12:X=20:GOSUB6000:PRINT"VELOCITY":X=50:GOSUB6000:PRINT"ALTITUDE
250 Y=13:X=20:GOSUB6000:PRINT"(FT/SEC)":X=50:GOSUB6000:PRINT"  (FT)"
260 Y=18:X=20:GOSUB6000:PRINT"ESTIMATED TIME TO LANDING "
270 Y=22:X=1:GOSUB6000:FORI=1TO79:PRINT"-";:NEXTI
275 Y=23:X=1:GOSUB6000:PRINT"0  "
280 FORI=1TO7:X=10*I:GOSUB6000:PRINTI;:NEXTI
290 X=30:Y=24:GOSUB6000:PRINT"ALTITUDE (X10,000 FT.)":GOSUB6000
310 VE=-100:MT$="               ":FU=10000:AL=80000:DE=5:BU=32
320 FORT=1TO10000
330 IFT/2=ING(T/2)THENPRINTCHR$(7);
340 VE=VE+((BU-32)*25E8)/(25E8+AL*AL))
345 VE=INT(VE)
350 AL=AL+INT(VE/2)
360 IFAL<0GOTO3000
370 IFFU<500THENGOSUB2000
380 FU=FU-BU/2
385 IFFU<=0THENFU=0:BU=0
390 IFBU<=0THENB$="NO BURN":GOTO410
400 B$=STR$(INT(FU/BU))
410 X=38:Y=5:GOSUB6000:PRINTMT$:GOSUB6000:PRINTB$
420 X=21:Y=9:GOSUB6000:PRINTBU:X=50:GOSUB6000:PRINTFU
430 X=22:Y=14:GOSUB6000:PRINTVE:X=50:GOSUB6000:PRINTAL
440 IFVE>=0THENA$="ESCAPE":GOTO460
450 A$=STR$(INT(AL/ABS(VE)))
460 Y=19:X=38:GOSUB6000:PRINTMT$:GOSUB6000:PRINTA$
461 TA=INT((AL+500)/1000):IFTA>80THENTA=80
462 IFTA<1THENTA=1
463 Y=21:X=TA+1:GOSUB6000
465 IFFU=0GOTO500
470 GOSUB5000:IFZ=13GOTO500
480 BU=12+4*(Z-48)
490 IFZ=48THENBU=0
500 FORTI=1TODE:A=SIN(10):NEXTTI
505 VP=VE:AP=AL
510 NEXTT
2000 FORJ=1TO2
2005 X=36:Y=12:GOSUB6000:PRINT"LOW FUEL"
2010 Y=13:GOSUB6000:PRINT"WARNING"
2020 A=SIN(10)
2030 GOSUB6000:PRINTMT$:Y=12:GOSUB6000:PRINTMT$
2035 A=SIN(10)
```

```
2040 NEXTJ
2050 DE=I:RETURN
3000 SP=(VP+VE)/2
3010 IFSP<-25GOTO3200
3015 PRINT:PRINT
3020 PRINTTAB(20)"CONGRATULATIONS, YOU TOUCHED DOWN AT A MERE "
3030 PRINTTAB(30)SP;" FT./SEC.  A SAFE LANDING !!!"
3040 PRINT:PRINTTAB(20)" DO YOU WANT TO TRY AGAIN AND"
3050 PRINTTAB(20)" ";:INPUT"PROVE IT WASN'T LUCK ";N$
3060 IFN$="N"THENRUN"BEXEC*"
3070 GOTO190
3200 PRINTCHR$(26)
3210 N=40
3220 FORI=1TON:X=1+INT(79*RND(I)):Y=1+INT(23*RND(1))
3225 GOSUB6000:PRINTCHR$(33+INT(15*RND(1)):GOSUB6000:NEXTI
3230 X=20:Y=10:GOSUB6000:PRINT"YOU JUST BLEW A CRATER,"
3240 Y=11:GOSUB6000:PRINTABS(VE);" FEET IN DIAMETER, ON THE
3250 Y=12:GOSUB6000:PRINT"SURFACE OF THE MOON.  BETTER TRY AGAIN...
3260 Y=14:GOSUB6000:INPUT" READY (Y/N) ";N$
3270 GOTO190
5000 Z=PEEK(64513)
5005 IFZ=13THEN RETURN
5010 IFZ>128THENZ=Z-128:RETURN
6000 PRINTCHR$(27);CHR$(61);CHR$(Y+31);CHR$(X+31);:RETURN
```

ASK THE DOCTOR - PART IV
GOOD NEWS/BAD NEWS

Robert M. Tripp, Ph.D.
The COMPUTERIST, Inc.
P.O. Box 3
So. Chelmsford, MA 01824

In last month's issue I announced that Synertek Systems has informed me of an improvement to the SYM monitor which should solve the audio cassette sensitivity problem that I had mentioned in several columns. I have since received a copy of the new SYM-1 Supermon Version 1.1 on a pair of EPROMs (which I had supplied to them) and have had some chance to evaluate the new version. The documentation I received was in the form of a two page letter. Not having the monitor listing limited by ability to fully evaluate the changes.

The Good News

According to the letter only two minor hardware changes are required in the cassette circuit. This are similar to some reported independtly by other users and reported in an earlier column. "Change C16 to .22 microfarad" and "change R97 to 1K ohm".

This list of improvements that accompanied the V1.1 monitor, along with my comments appears below. (The Synertek notes are in bold face. My comments are normal type.)

1. The **Improved High Speed Cassette** read/write is significantly better than before. I was able to write and read quite constantly and was able to produce a tape on one type of recorder and read it on another. The volume/tone range was much wider. Whereas before you had to be right on for any chance of success, now you can have a reasonable variation in volume and tone and still get a good read. This is particularly important when you are using different recorders with different characteristics. The two recorders I tested with were a Superscope C-190 and a Pioneer Centrex. These fairly high quality recorders have **not** worked reliably with the old V1.0 monitor. A suggestion I had made to Synertek back in June 1978 was to make the leader time variable. While the 8 seconds they had built-in in V1.0 is acceptable when you are only occasionally storing a program, it was much to long if you intended to use the tape service to save small chunks of data - mailing list information for example. The above note says that the leader time is now maintained in ram and can be changed by the user if necessary. Since I did not have the listing or additional information, I was not able to test this out. But, assuming it does work, this can be a very significant improvement. some programs I have written require a lot of extra code simply to get around the "fixed" leader problem. They should be much simpler now, since I should be able to set the leader time in ram and then use the tape cassette routines directly.

2. **KIM read. Read routine improved.** This has been one of the biggest problems for the SYM-1 since it release. The V1.0 monitor had a simple, but powerful, bug. It made an invalid test for the KIM format "end-of-data" character, and treated the legal 32 46 ASCII pair as an ASCII "/", thereby terminating prematurely whenever it encountered a "2F" in the data. This made the KIM format mode of the SYM-1 essentially useless. This has been fixed in the new version. This means that it is now possible to distribute software, data bases, source files, etc. between the KIM-1, SYM-1 and AIM 65 using the common KIM format.

3. **Beeper frequency adjusted for maximum output.** I'll take their word for this. It does sound a little louder, but then I had never had any trouble with the beeper in V1.0

4. **During the VERIFY command a BREAK key will stop printout without printing an error message.** I didn't test this minor improvement, but it is nice to keep error messages for real errors.

5. **BREAK key is looked for on current loop interface.** If you are using a teletype device, it is handy to have the BREAK key work, so this change is definitely good.

6. **Log-on changed to SY1.1.** Yes.

7. **After paper tape load the error message count is displayed.** I do not have any paper tape facility to test this, but it is a minor improvement.

8. **Ability to return to a higher lever program (left arrow).** I do not quite understand what this is supposed to mean, but I am sure when additional documentation is available it will make sense.

9. **Cassette file I.D. displayed on left digit seven segments.** This is both cute and useful. They have simply taken the ID value and put it out on the leftmost digit. It does take a bit of deciphering though. The figure below shows the value of each segment on the display. These must be separately read and then added together to get the file ID. It is useful when you are searching the tape for a particular tape ID.

10. **Unwrite-protect routine added to cassette logic.** Again, I could not test this due to zero documentation.

11. **Register name improvement on display during R command.** Hooray! Now the display shows the register name, not a "hard-to-remember-and-interpret" arbitrary number to identify which register you are examining. P for program counter; S for stack; F for flags; A for A register, to represent an X for the X register; and Y for the Y register. A simple but very nice improvement.

12. **Debug-on will not cause ram to be write protected.** I did not test this, but it sounds reasonable.

That's the good news.

The Bad News

The bad news isn't all that bad, but should be considered. First, the changes to the Supermon do move some code around and change some "internal" entry points. Although the Synertek programmer I talked to said that this was not going to be very important since the main entry points were not touched, I found the first program I tried to run, the SYNC generator from the Reference Manual, would not work since two of the routines it requires have moved. How great a problem will this be? It is difficult to guess. I haven't seen the listings and do not know what routines were changed and also do not know how often other programmers have used them directly. It will be a problem for anyone who is trying to make program for distribution since there may be a requirement for two versions - one for V1.0 and another for V1.1 - and this adds to the expense and can cause distribution problems. Hopefully, the number of routines affected is small and isn't a big problem - but at present, "Who knows?".

Second, the V1.1 does use up some (most?, all?) of the Scratch Pad RAM in the System RAM. While this is not necessarily a big problem for future programs, it may cause problems for existing programs which use this previously available resource. Care will have to be taken when transferring programs from V1.0 to V1.1 to take this change in scratch pad availability into account.

Third, Synertek does not seem to have a policy yet for how the new V1.1 will be distributed. They are still waiting for feedback from myself and a couple of other users before committing to ROM, so it will be some time before any of the V1.1 are available at all. Then there is the question of systems already in the field or on dealer's shelves. Will there be a reasonable "exchange" policy, say Synertek's actual ROM production cost of $10-$15.00, or is some outlandish price going to be charged. I strongly feel that Synertek has the responsibility to offer the new V1.1 at the lowest price possible. Some of the changes they have made are not "cosmetic" or simple "improvements". They are basic "corrections" to their original "flawed" V1.0.

SYM-1 Codes

Ever wonder what the various codes were that the SYM used: keycode, ASCII code, and display code? You can look them up in the SYM manual in various places, but, why not let the SYM itself generate a display of these codes. The following program is an aid in establishing the relations between the three different codes. Start the program at 0000. The display goes blank, and when a key is depressed, the display will show key code, ASCII and display-scan code for a short time, and go blank again with a "beep".

Submitted by
Jan Skov
Majvaenget 7
DK-6000 Kolding
The Netherlands

```
                      SYM-1 CODE DISPLAY
                      JAN SKOV
                      FEBRUARY 1979

0000                         ORG    $0000

                      SYM SUBROUTINES

0000                  ACCESS *      $8B86  SYSTEM RAM ACCESS
0000                  SPACE  *      $8342  OUTPUT SPACE TO DISPLAY
0000                  INCHR  *      $8A1B  INPUT CHARACTER
0000                  OUTCHR *      $8A47  OUTPUT CHARACTER
0000                  OUTBYT *      $82FA  OUTPUT BYTE
0000                  SCAND  *      $8906  SCAN DISPLAY
0000                  BEEP   *      $8972

0000 20 86 8B  START  JSR    ACCESS
0003 A2 06            LDXIM  $06
0005 20 42 83  LOOP   JSR    SPACE
0008 CA               DEX
0009 D0 FA            BNE    LOOP
000B 20 1B 8A         JSR    INCHR
000E 85 EF            STAZ   $00EF
0010 A9 2D            LDAIM  $2D
0012 20 47 8A         JSR    OUTCHR
0015 A5 EF            LDAZ   $00EF
0017 20 FA 82         JSR    OUTBYT
001A AD 42 A6         LDA    $A642   DISPLAY BUFFER
001D 20 FA 82         JSR    OUTBYT
0020 A2 0B            LDXIM  $0B
0022 86 EE            STXZ   $00EE
0024 86 ED            STXZ   $00ED
0026 20 06 89  LOOPA  JSR    SCAND   DISPLAY AND
0029 C6 ED            DECZ   $00ED   TIMER LOOP
002B D0 F9            BNE    LOOPA
002D C6 EE            DECZ   $00EE
002F D0 F5            BNE    LOOPA
0031 20 72 89         JSR    BEEP
0034 4C 00 00         JMP    START
```

THE MICRO SOFTWARE CATALOG: VIII

Mike Rowe
P.O. Box 3
S. Chelmsford, MA 01824

Name: **Missile-Anti-Missile**
System: **Apple**
Memory: **16K**
Language: **Apple II Soft**
Description: Simulated missile attack on 3-D Map of USA
Copies: **30**
Price: **$9.95 + $1.00** postage & handling
Includes: Cassette with instructions
Author: **T. David Moteles & Neil Lipson**
Available from:
Progressive Software
P.O. Box 273
Plymouth Mtg., PA 19462

Name: **DISK DUMP/RESTORE**
System: **Apple II with disk**
Memory: **32K (min)**
Language: **Applesoft II and machine language**
Hardware: **Apple II, Disk II**
Description: A disk-tape utility to dump and restore all Integer, Applesoft II, and Binary programs automatically. The program names. Binary program addresses, and all commands necessary to re-load the programs from tape and restore them again to disk under their original names are stored on tape header file.
Copies: **Just released**
Price: **$8.00**
Includes: Cassette and instructions
Author: **Alan G. Hill**
Available from:
Alan G. Hill
12092 Deerhorn Dr.
Cincinnati, Ohio 45240

Name: **NOT ONE**
System: **KIM**
Memory: **1K**
Language: **Assembly**
Hardware: Bare Kim!
NOT ONE is an exciting, fast moving game of skill, strategy, and change for one to five players (including KIM). The game is designed for use with KIM's onboard display and hex pad.
Besides being an entertainment game, the NOT ONE package was designed to introduce some powerful general-purpose output manipulation subroutines for the KIM's LED display. These include variable-speed, scrolled alpha-numerics!
The manual also discusses LED segment codes in an effort to increase the user's knowledge of the display.
Author: **Steven Wexler**
Price: **$15.00**
Includes: Source listing, manual, and cassette
Available from:
SJW, Inc.
P.O. Box 438
Huntingdon Valley, PA. 19006

The 6502 Program Exch.
2920 Moana
Reno, NV. 89509

Name: **A Forth System**
System: **Apple II**
Memory: **24K or Larger**
Language: **40% ASSEMBLY, 60% Forth**
Hardware: **Disk II**
Description: A unique software package for software buffs and serious programmers who have gotten tired of programming in integer basic and machine language. FORTH is an extensable language, allowing the programmer to "define" new dictionary entrys that use previous entrys. Most of FORTH is written in FORTH. Benchmarks show that FORTH executes 20 times faster than BASIC.
Included in the package are:
1) Powerful screen editor for system development.
2) Decompiler - used to generate to some extent a source listing. It can be used to list our portions of FORTH itself.
3) Utility package - dump, disk maintenance etc. does not use apple II dos.
4) Completely documented using a special disk retreival system. includes some programming examples. Editor, decompiler is available on source.
Copies: **Just Released**
Price: **$39.95** + tax for california residents
Includes: One mini diskette + manual
Author: **John T. Draper**
Available from:
Captain Software
PO Box 575
San Francisco, CA 94101

Name: **Function Graphs and Transformations**
System: **Apple II**
Memory: **16K minimum if Applesoft is in ROM, otherwise 32K minimum**
Language: **Applesoft (floating point Basic)**
Hardware: **No special hardware**
Description: This program uses the Apple II high resolution graphics capabilities to draw detailed graphs of mathematical functions which the user defines in Basic syntax. The graphs appear in a large rectangle whose edges are X and Y scales (with values labeled by up to 6 digits). Graphs can be superimposed, erased, drawn as dashed (rather than solid) curves, and transformed. The transformations available are reflection about an axis, stretching or compressing (change of scale), and sliding (translation). The user can alternate between the graphic display and a text display which lists the available commands and the more recent interactions between user and program. Expected users are engineers, mathmaticians, and researchers in the natural and social sciences; in addition, teachers and students can use the program to approach topics in (for example) algebra, trigonometry, and analytic geometry in a visual, intuitive, and experimental way which complements the traditional, primarily symbolic orientation.
Copies: **Just released**
Price: **$14.95** (Cat. No.: AHE0123)
Includes: cassette tape, 12-page instruction booklet
Author: **Don Stone**
Available from: many computer stores or
Powersoft, Inc.
P.O. Box 157
Pitman, NJ 08071
(609) 589-5500

Name: **6502 VDR**
Systems: Any 6502 with room available at $200 or $DD00
Memory: **½K**
Language: **6502 machine code**
Hardware: **Memory-mapped video board such as Polymorphic Systems VTI, Solid State Music VB-1B, Etc.**
Description: Organizes memory-mapped display for teletype-like use including automatic scrolling, line wrap-around, clear screen commands, etc.
Copies: **30**
Price: **$9.50 plus $1 shipping**
Includes: Operating Manual, detailed configuration information, and complete commented source listing.
Order: Package includes KIM compatable tape cassette with both $200 and $DD00 versions included. Charge cards, phone and mail order accepted.
Available from:
Forethought Products
97070 Dukhobar #D
Eugene, Oregon 97402

Name: **CHEQUE—CHECK™**
System: **PET**
Memory: **8K**
Language: **BASIC, with machine language subroutine**
Hardware: **PET 2001-8 (or 2001-16/32 on special order)**
Description: CHEQUE-CHECK reduces the probability of error in reconciling bank statement and checkbook, even for those experienced in the art. More important it greatly reduces the time required to find and correct an error when one does occur, because it "remembers" individual entries for later review and modification if necessary. Designed and tested for ease of use, CHEQUE-CHECK is suitable for novice or expert, and requires no tape files or knowledge of programming. Reviewed in May 1979 issue of Robert Purser's Reference List of Computer Cassettes.
Copies: **60 sold** in first three months.
Price: **$7.95** (quantity discount available)
Includes: Cassette in Norelco style box, Description and operating instructions, zip-lock protective package.
Designer: **Roy Busdiecker**
Available from: Better computer stores or directly from
Micro Software Systems
P.O. Box 1442
Woodbridge, VA 22193

Name: **Disk Catalog Program**
System: **Apple II**
Memory: **32 K minimum**
Language: **Integer Basic and Machine Language**
Hardware: **Apple II, DISK II**
Description: This program consists of two modules. The first, DCATPRO, is a general purpose data base catalog program for books, records, tapes, programs on diskette, etc. Features include 40 col. records, 5 fields (2 with adjustable length), and super fast machine language sort. The second, GENCPINP, **automatically** processes any set of Apple II diskettes and generates a data base for DCATPRO by reading the D$CATALOG information for each diskettes. Then you know what you have and **where it is,** all without having to type in a lot of data.
Copies: **Over 100 sold**
Price: **$10.00 postpaid**
Includes: Programs on cassette and 5 pages of documentation
Arthur: **George W. Lee**
Available from:
George W. Lee
18003 S. Christina Ave.
Cerritos, California 90701

Name: **Generalized File Management**
System: **APPLE II**
Memory: **16K**
Language: **Integer Basic**
Hardware: **APPLE II, DISK II**
Description: This package allows you to create, update, and print disk files. The names of fields and files, number of fields, individual field lengths, and file size is user defined. You can decide what headings you want to see (if any) when you print or display and record or the entire file. You can use this package to create such files as: Parts lists, phonenos., List of birthdates, name and address, and whatever...
Copies: **Just released**
Price: **$16.50**
Includes: Diskette that contains two programs, some sample file useages (birthdates, parts list), and a user manual.
Author: **Lee Stubbs**
Available from:
Les Stubbs
23725 Oakheath Pl. Harbor City, Ca 90710

Name: **WEAVER**
System: **Apple II**
Memory: **32K**
Language: **Integer Basic**
Hardware: **Disk II**
Description: WEAVER simulates as multi-harness loom with control of warping, hook-up and treadling. Weaving drafts of 40 threads of warp and 40 threads of weft are drawn in 15 colors for patterns requiring up to 24 harnesses. Weaving patterns are saved and called by name from disk storage. The user-interface is designed for easy and efficient use by a weaver. Nine pages of documentation include a glossary of commands which defines the functions of the program and a sample draft with descriptive data entry.
Copies: **New program.**
Price: **$15.00** on cassette tape, **$20.00** on diskette with five sample drafts.
Author: **Bruce Bohannan**
Available from:
Bruce Bohannan
2212 Pine Street
Boulder, CO 80302

Name: **Address and Perpetual Calendar**
System: **APPLE II**
Memory: **32K**
Language: **Applesoft II**
Hardware: **APPLE II w/Disk II**
Description: This program maintains your master address file on disk. User follows a master menu to add or change names, look for specific names or review entire file (or part) name by name. All outputs are formatted. Look and change records with a search function i.e., If you do not remember how to spell a name then enter the number of letters you do know and the program will walk you through all names beginning with what you entered until you find the one you want. A birthday function is included that will search your entire file and list all names, birthday and age for any given month. A special feature loads up a Perpetual Calendar program that will display any month (formatted) between the years 1704 and 2099 and highlights any particular day. Return to address program is optional.
Copies: **Just released.**
Price: **15.00 ppd**
Includes: Disk and instructions
Author: **Edward S. Kleitches**
Available from:
Edward S. Kleitches
7207 Camino Grove
San Antonio, Texas 78227

INSIDE THE KIM TTY SERVICE

Ben Doutre
621 Doyle Road
Mont St-Hilaire, Quebec
Canada J34 1M3

The fact the KIM's serial TTY port, plain and unmodified, will operate comfortable at 9600 bauds does not seem to be widely known. I, for one, went the parallel interface route as soon as I acquired a higher speed terminal, and I suspect that many others may have done likewise. After all, what can one expect of an interface described in the User's Manual in these terms: "You are not restricted to units with specific bit rates (10 CPS for TTY) since the KIM-1 system automatically adjusts for a wide variety of data rates (10 CPS. 15 CPS. 30CPS. ETC.(.· "That's pretty wide, alright, from 10 to etc. Other writers have been equally vague. Gary Tater in MICRO 9:14, "A Fast Talking TIM" mentions that "KIM can adapt to terminal frequencies up to 2400 baud...". This was the last straw, and I either had to pull the plug on my "Fast Talking KIM". or attempt to put the record straight

First off, let me say that according to my interpretation of what goes on in KIM, the theoretical maximum baud rate of the TTY port is 15,625. How's that for pinning down the etc? Not that you should try to operate at this rate without some of the well-known "fine tuning", but there is no reason why you can't hook up your 9600 or 4800 baud terminal, with 30 cents worth of gates, and be up and running, with or without reading the following details. If you want to know from whence this bonanza, here is the story.

The smarts for the KIM TTY interface are in the monitor software, so let's start at that end. There are two main TTY I/O routines: GETCH at 1E5A and OUTCH at 1EA0. GETCH returns with the character in A but strips off the parity bit in the process. If you need bit 7 (counting from 0) for your own deep, dark reasons, then retrieve the full character from CHAR at OOFE on your return. OUTCH (love that label!) outputs a stop bit, then a start bit, then 8 data bits (LSB first), then another stop bit. It may seem illogical to start with a stop, but remember that, aside from slow machinery, the main purpose of a stop bit (line high) is to make sure that the start bit (line low) will be recognized. In any case, the stop interval is 2 bits long plus the delay between calls to OUTCH.

Both GETCH and OUTCH are timed by subroutine DELAY at IED4. (GETCH also used DEHALF to move its strobe to the mid-point of a bit interval, but let's not get technical.) DELAY does its thing based on the contents of a 16-bit counter named, for some obscure reason, CNTH30 (high byte, at 17F3) and CNTL30 (low byte, 17F2). If this counter is equal to 0000 or less, DELAY falls through all the way, with a resulting minimum bit time of 64us. (Let's assume your crystal is bang-on 1 MHz.) Presto: devide 64us into a million, and you come up with 15,625 baud.

Not convinced? OK, here's more. Every time we add one to the counter, DELAY adds another 14 us to its timing loop. The high end of the baud scale looks like this:

Counter	Bit Time (us)	Baud Rate
0000	64	15,625
0001	78	12,820
0002	92	10,869
0003	106	9,434
0004	120	8,333

If we turn this around and start with some of the usual standard baud rates, we can calculate the bit times and counter values required. For instance, 9600 bauds obviously needs something betweem 2 and 3. DELAY doesn't do fractions - it doesn't even like odd numbers. And how does the counter get properly loaded anyway?

We've left the best to the last, a little jewel called DETCPS at 1C2A. DETCPS is entered following a system reset with TTY enabled. Its brief hour of glory is in measuring the duration of the start pulse of the first character you feed in after a Reset. It quickly stuffs the results in the 16-bit counter, then goes out for coffee until the next Reset. The question is: will DETCPS buy 9600 bauds? The answer is YES. albeit a little reluctantly. The thing is the DETCPS is sampling the input port, waiting for the line to go low - it checks for this every 9 us, so it could miss your start pulse start by this much. Once the line is low, it squirrels away 14 us counts, checking for line high every 14 us. So it could miss the end of your start pulse by 14 us.

At 10, 15, 30 or etc CPS. this sloppiness is probably acceptable. With a Model 33 on the line, DETCPS gaily reports 02C2 plus/minus OB, for instance. But if it comes up with 0004 instead of 0003 at 9600 bauds, your TV screen will give you a reasonable facsimile of a Chinese fortune cookie slip. Just look at it as another Butterfield game - Reset-Delete-Reset-Delete-Reset-Delete BINGO! Anyway, how many times a day do you Reset? Once you get that 3, your link with KIM will be rock solid.

There are a number of facinating details, but I will spare you the pyrotechnics. If all this is on the leve, I should be able to prove it, right? Well, I have an ESAT-100 (RHS Marketing) video board equipped with an AY3-1015 UART hooked up to the KIM TTY port. The manual admits to a -1% to DETCPS. I set the speed selector switch to each of the 6 rates available, did 10 resets at each and recorded the counts. (A clever piece of programming, at that!) Except for 9600, all resets were OK the first time around. The counts did not vary, except for 300 baud. The results look like this:

Baud Rate	Bit Time (us)	Calc. Count	Meas'd Count
9600	104.2	0003	0003
4800	208.3	000A	000B
2400	416.7	0019	001A
1200	833.3	0037	0038
600	1666.7	0072	0074
300	3333.3	00EA	00EC/00ED

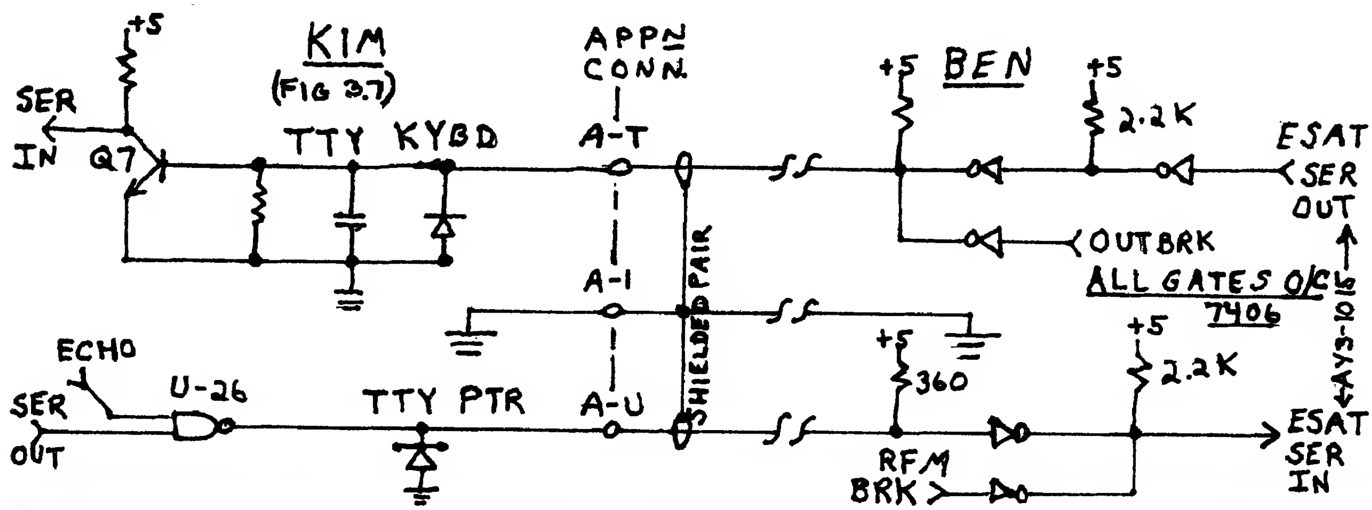
A few further words of explanation for the fellow who may be hung up because he has been spared intimate relations with "real" TTY machines. (You experts can go figure out an algorithm or two - try infinite recursion on "Every rule has an exception, except this one.")

Referring to the KIM-1 User's Manual, Fig. 3.7, you will see two KYBD lines and two PTR lines. The action at the other end of these lines is assumed to be as follows: - During idle conditions, the keyboard lines are shorted out, generating a continuous high at the input to Q7; the printer lines are connected to a "selector magnet" (quaint) or a relay which is drawing a nominal 20 mA. -when the keyboard is sending characters, the KYBD lines are open-circuited for zero bits and shorted for one bits. When KIM sends characters on the PTR lines, it opens the circuit for zero bits by floating the output of O/C gate U26 (7438), and closes the circuit for one bits by pulling U26 to ground. Incidentally, this 7438 can sink up to 48 mA.

If you want to simulate this hardware with some other device, you need to feed the line labelled "TTY KYBD" with positive logic signals (low for ones, open for zeros) from the line labelled "TTY PTR". You should note that the keyboard line has a 220-ohm pull down resistor on it, and that the printer line has no pull-up.

You may also notice, if your terminal has a FDX/HDX selector switch or jumper, that the FDX no longer works as advertised. This is just KIM trying to be helpful, with a wired-in interconnect which echos received characters on the output line. If this keeps you awake at night, cut the trace between pin 11 and U15 and pin 10 of U26, and connect pin 10 of U26 to Vcc. (I haven't tried it, but it should work. I'm a sound sleeper.)

If you need a for-example, I show a diagram of my own interface logic, based on a 7406 gate package, which is working quite satisfactorily. There are probably 1000 other ways of doing it, each one of which can be improved by SuperSilicon. If it works and doesn't smoke, have at it.



THE INTEGER BASIC TOKEN SYSTEM IN THE APPLE II

Frank D. Kirschner
2643 Rockledge Trail
Dayton, OH 45430

There are two primary methods of storing BASIC programs in microcomputers. One involves storing the entire program, letter by letter and symbol by symbol somewhere in memory, and interpreting the ASCII codes on execution. This is typical of BASIC compilers and some interpreters, like the TRS-80 Level 1. A more memory-efficient system uses tokens, eight bit bytes each of which represent a BASIC word or symbol. The TRS-80 Level II uses this method, as does the Apple II, to which the examples which follow apply.

When in Integer BASIC, the Apple stores characters as they are entered in a character buffer (hex locations 0200 to 02FF). When "return" is entered, BASIC "parses" the entry (that is, interprets the ASCII characters and breaks the instruction into executable parts). It determines what is a command, what are variables, data and so forth. If it is legal and is preceded by a number between 0 and 32767 (a line number), it stores it in memory in a fashion discussed below. If there is no line number, it simply executes the command and awaits further instructions.

The way the programs are stored is quite clever. When BASIC is initiated (control B or E000 G from the monitor) several things happen. First, the highest available user memory (RAM) is stored in memory locations 004C (Lo byte) and 004D (Hi byte), called the HIMEM pointer. Also, locations 00CA and 00CB, the start-of-program pointer, get the same numbers, since there is no program as yet. As program steps are entered, they are stored starting at the top of memory, highest line numbers first, and the start-of-program pointer is decreased accordingly. See Figure 1. When a line with a higher number than some already in memory is entered, they are shuffled to preserve the order. One application: if you enter a program and then hit control B, the program is **not** scratched (or erased); only the start-of-program pointer is affected. Since powering up the Apple fills the memory with a pattern of ones and zeros (it looks like FF FF 00 00 ...) from the monitor, it is easy to find the start of the program and then manually reset CA and CB to that location.

This is the way program instructions are stored in memory: (All numbers are in hex)

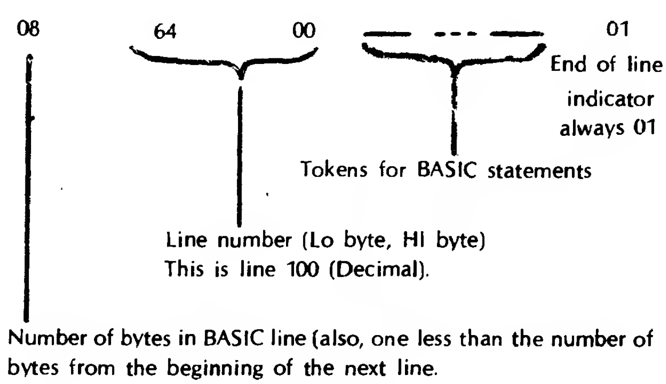


Figure 2

As an example, power up the Apple, bring up BASIC, and enter

100 PRINT 0,50

Enter the monitor (by pushing "reset"), and then examine the program by entering

EXAMPLES FOR
16K Apple

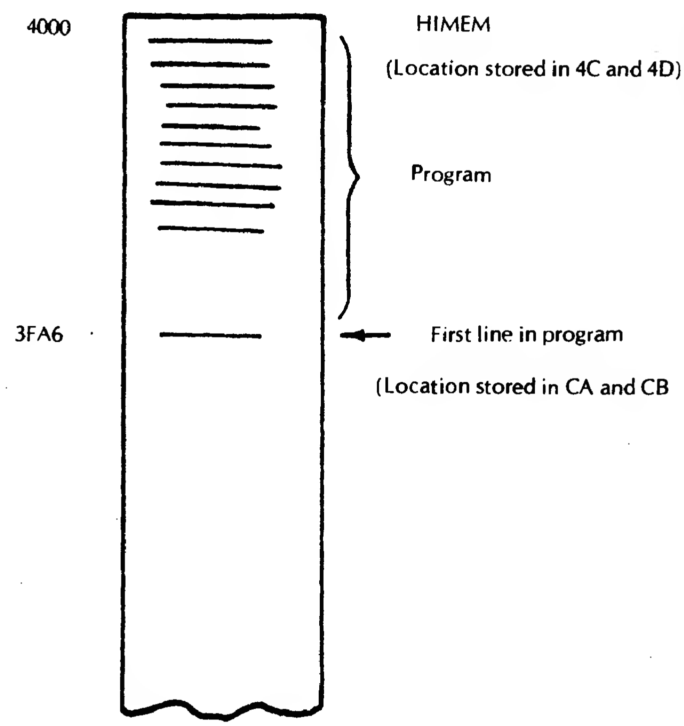


Figure 1

Memory Map for Program Storage

3FF4.3FFF return

(Locations for a 16K Apple. Subtract 2000 hex for a 4K or add 4000 hex for a 32K Apple.) You will see this:

3FF4 - 0C 64 00 62
3FF8 - B0 00 00 49 B5 32 00 01

which means:

0C	There are 12 bytes in this line
64 00	It is line 100 (Decimal)
62	PRINT (see Table 1 for complete list of tokens
B0	The next two bytes are a number (rather than tokens)
00 00	The number 0
49	The comma in a PRINT statement
B5	Another number follows
32 00	The number 50
01	End of BASIC line

To demonstrate the use of this information, return to BASIC and try to enter the following BASIC line:

100 DEL 0,50

You will get a syntax error, because the Apple Interpreter does not allow the command DEL in deferred execution mode. Now do this: reenter the monitor and change the 62 (PRINT) to 09 (DEL) and the 49 (,for PRINT) to 0A (, for DEL) by entering

3FF7: 09 Return

3FFB: 0A Return

Reenter BASIC (control C) and list. Try this instruction by adding lines between 0 and 50, running the program, and then listing it. This allows you to write a program which will carry out some functions only the first time it is run and then automatically delete those lines.

In addition to inserting instructions which cannot be entered as deferred commands, you can modify the program under program control. As an example, here is a program which will stop and start listing a long program by hitting a key on the keyboard.

Bring up BASIC.
Enter: 257 LIST 0: RETURN
HIT RESET, 3FF6.3FFF RETURN
You will see
3FF6 - 0A 01
3FF8 - 01 74 B0 00 00 03 5B 01
What this means:
3FF6: 0A Ten bytes in line
3FF7,8: 01 01 LINE 257
3FF9: 74 TOKEN FOR LIST
3FFA: B0 Means "Number follows"
3FFB,C: 00 00 LINE TO BE "LISTED" (LO, HI)
3FFD: 03 TOKEN FOR COLON
3FFF: 01 End of BASIC LINE
Now enter 3FF7: FF FF Return
Cont. C, List
You have 65535 LIST O: RETURN
Now enter
100 X=PEEK (-16384): POKE -16368, 0:1F
X 127 THEN 0: GOTO 100
Reset, 3FCF.3FFF Return
Change line no. from 100 to 65534 by entering 3FDO; FE FF Return
Change GOTO 100 to GOTO 65534 by entering 3FF3: FE FF
Change the 0 in "THEN 0" to 65533 by entering 3FEE: FD FF
In like manner, enter these remaining steps: (Under each number which has to be entered through the monitor, the Hex equivalent, in reverse order as it must be entered, appears)

```
65533  I = I  PEEK (I): IFI> PEEK  (76)+
(FD FF)
               256*PEEK (77) THEN END: GOTO
               65531
               (FB FF)
65532 X=PEEK (-16384):POKE -16386,0:
(FC FF)
          IF X  127 THEN 65534
                           (FE FF)
65531 POKE 16374, PEEK (I +1): POKE 16380
(FB BB)
           PEEK (I+2): GOSUB 65535
                              (FF FF)
32767 I=PEEK (202)  256* PEEK (203)
```

The steps must be entered in reverse order (i.e descending line numbers) because the interpreter orders them by their number when entered, and will not re-order lines when the numbers have been changed through the monitor.

The reason for making all these line numbers very high is so the applications program will fit "under" the list program.

Now, in the monitor, move the start of program and HIMEM pointers below the program:

3A: 49 3F Return

4C: 49 3F Return

Hit control C and list. Nothing is listed. The program has been stored in a portion of memory temporarily inaccessible to BASIC. Load your applications program, make sure all the line numbers are less that 32767, and change HIMEM through the monitor (4C: 00 40) and execute RUN 32767. The program will list until you hit a key and then resume when you hit a key again. It uses the fact that each line begins with the number of bytes in the line followed by the line number. Numbers of successive lines are found and "POKE into the appropriate location in line 75535, which then lists each line.

Using these methods you can exercise considerably more control over the BASIC interpreter in your microcomputer.

TABLE I

APPLE II INTEGER BASIC TOKENS

BASIC COMMAND OR FUNCTION	HEX TOKEN	
ABS	31	
(	3F	
)	72	
ASC (	3C	Includes left paren.
)	72	
"	28	first quote
"	29	second quote
AUTO	0D	
,	0A	
CALL	4D	
CLR	0C	
COLOR=	66	Includes =
CON	60	
DEL	09	
,	0A	
DIM	4F	Numeric Arrays
(	34	
)	72	
DIM	4E	String Array
(	22	
)	72	
$	40	
DSP	7C	Numeric Variable
DSP	7B	String Variable
END	51	
FOR	55	
=	56	
TO	57	
STEP	58	
GOSUB	5C	
GOTO	5F	
GR	4C	
HIMEN:	10	Includes :
HLIN	69	
	6A	
AT	6B	
IF	60	
THEN	24	When followed by a line no.
THEN	25	When followed by GOSUB or a basic operation
INPUT	54	Numberic Variable
INPUT	52	String Variable
INPUT	53	Input if followed by ...
,	27	
"	28	first
"	29	Second
IN #	7F	Includes #
LEN (	3B	Includes (
LET	5E	
LIST	74	
	75	

BASIC COMMAND (CONT)	HEX TOKEN	
LOAD	04	
MAN	0F	
NEW	0B	
NEXT	59	
	5A	
NO DSP	79	
NO TRACE	7A	
PDL	32	
(	3F	
)	72	
PEEK	2E	
3F	(	
72	)	
PLOT	67	
,	68	
POKE	64	
,	65	
POP	77	
PRINT	63	If used alone
PRINT	62	Numeric Variable
:	46	
,	49	
PRINT	61	String Variable
"	28	First
"	29	Second
PR #	7E	Includes #
REM	5D	
RETURN	5B	
RND	2F	
(	3F	
)	72	
-	36	
SAVE	05	
SCRN (	3D	Includes (
,	3E	
)	72	
SGN	30	
(	3F	
)	72	
TAB	50	
TEXT	4B	
TRACE	7D	
VLIN	6C	
,	6D	
AT	6E	
VTAB	6F	
:	03	
=	71	In assignment
AND	1D	
OR	1F	
MOD	1F	
NOR	DE	

PROGRAMMING THE 6502
by Rodney Zaks

Reviewed by
John D. Hirsch
Berme Road
Kerhonkson, NY 12446

In the introduction to this book the author tells us it can be used by a person who has never programmed before. Chapter one does begin with a clear presentation of some basic techniques, such as binary arithmetic. But the quality of the book rapidly degenerates in succeeding chapters, which read as though they had been assembled from manufacturer's literature and other sources, with more help from a paste-pot than a pencil.

The quality of the writing is technical-manualese and the illustration have the same mechanistic flavor. Also the illustrations and writing are sometimes only tenuously related.

A novice programmer would probably give up along about Chapter 3, when assembly language routines are introduced even though assembly language is not explained until near the end of the book.

The organization of the book has a certain random quality. For instance, integer addition, subtraction and multiplication are explained in some detail in the chapter on basic programming techniques, and then division is relegated to one paragraph while the chapter goes on to a very general explanation of subroutines.

The experienced programmer will not find the book very helpful either. A good chunk of the book is taken up by reprinting 6502 instructions, one per page, and potentially valuable chapters--such as the one covering 65'xx interfacing chips--are very perfunctory. Dr. Zaks has the annoying habit of constantly referring the reader to manufacturer's data sheets for more details

Chapter 9, covering data structures, is particularly puzzling. It covers data structures in a general way, with practically no information on how they can be implemented in 6502 assembly language. Perhaps the author intended this chapter for one of his other introductory computer books and pasted it in this one by mistake.

The publisher of this book has produced a good many other books which were either authored or co-authored by Dr. Zaks, all in a remarkably short time. Reading this book, it's easy to see how the trick is done.

The 6500 family software manual and Caxton C. Foster's charming introductory work PROGRAMMING A MICROCOMPUTER: 6502 (Addison-Wesley) are still the best texts for learning to program in 6502 machine or assembly language.

Chuck Carpenter
2228 Montclair Place
Carrollton, TX 75006

Renumbering Applesoft programs suddenly became possible. The resequence program in Jim Butterfield's "Inside Pet BASIC," (MICRO 8:39) solved the problem.

After clearing up a minor problem in the program (with help from Jim) I tried it on a 200 line program. Because of the way I started numbering in the first place, I had to fix-up about a dozen lines. But, I never would have gotten through that much renumbering otherwise.

As Jim mentioned in his letter to me, a machine language program would have ran a whole bunch faster. With DOS and having to find a place to locate such a program, the BASIC approach may be easier.

Here are some comments on the Applesoft version shown in Listing 1:

-Line 60005 has some prompting inputs to set-up the program.
-Use RUN 60005 to start renumbering.
-Line 60060 brances to a DELete line.
-Line 60160 is changed to a point to the line no. in Applesoft (2049 or $801).
Note: These are the pointers for Applesoft ROM
-Line 60160 was also changed to allow starting at any line number (M=LN-IN).
-Line 60170 changed to allow any numbering increment (M=M +IN).
-Line 60220 - tokens changed for Applesoft (this information is in the Applesoft II manual).
-Line 60260 and 60270 added to delete the renumber program and end it.

To make using the program easier, an append program (also for ROM) does the job. The assembly language program shown in listing 2 links the two programs together. You only need to do this if you want to renumber an existing program. (You can still load the renumber program before you start a new program.) Here's how you use it.

-Load the append program first. It fits in page 3 starting at $3A5.
-Load the lower line no. Applesoft program.
-Type Call 933 and (return).
-Load the higher line no. renumber program.
-Type CALL 955 and (return).
-Use RUN 60005 to start renumbering.

Be sure to record any output that appears on the screen. Write down the information and check the renumbering on the lines indicated. Putting longer line numbers in short spaces will be one message. Another will ask you to check where you used a THEN for a GOTO. The renumber program is not sure if it should renumber a line or a parameter.

My thanks to Jim Butterfield for providing us with such a useful program (and helping me get this one running). Also, thanks to Bob Matzinger from the Dallas Area Apple Corps for some modification suggestions and the Applesoft ROM append routine.

```
*
*3A5L

03A5-   A5 67      LDA   $67
03A7-   85 06      STA   $06
03A9-   A5 68      LDA   $68
03AB-   85 07      STA   $07
03AD-   38         SEC
03AE-   A5 69      LDA   $69
03B0-   E9 03      SBC   #$03
03B2-   85 67      STA   $67
03B4-   A5 6A      LDA   $6A
03B6-   E9 00      SBC   #$00
03B8-   85 68      STA   $68
03BA-   60         RTS
03BB-   A5 06      LDA   $06
03BD-   85 67      STA   $67
03BF-   A5 07      LDA   $07
03C1-   85 68      STA   $68
03C3-   20 F2 D4   JSR   $D4F2
03C6-   60         RTS
03C7-   FF         ???
03C8-   FF         ???
*
```

Listing 2

Applesoft append program. This program can be used to append any two programs together.

```
]LIST

60000  END
60005  HOME : PRINT : PRINT "RENUMBER:": PRINT : I
NPUT "FIRST LINE # - ";LN: PRINT : INPUT "INCREMEN
T - ";IN
60010  LET T = 0: DIM V%(100),W%(100): GOSUB 60160
: FOR R = 1 TO 1E3: GOSUB 60210
60020  IF G THEN  GOSUB 60090: NEXT R
60030  GOSUB 60160: FOR R = 1 TO 1E3:N =  INT (M /
 256): POKE A - 1,M - N * 256
60040  POKE A,N:V = L: GOSUB 60070:W%(J) = M: GOSU
B 60170: IF G THEN  NEXT R
60050  GOSUB 60160: FOR R = 1 TO 1E3: GOSUB 60210:
 IF G THEN  GOSUB 60110: NEXT R
60060  PRINT "*END*": GOTO 60260
60070  LET J = 0: IF T <  > 0 THEN  FOR J = 1 TO T
: IF V%(J) <  > V THEN  NEXT J:J = 0
60080  RETURN
60090  IF V <  > 0 THEN  GOSUB 60070: IF J = 0 THE
N T = T + 1:V%(T) = V
60100  RETURN
60110  GOSUB 60070: IF J = 0 THEN  RETURN
60120 W = W%(J): IF W = 0 THEN  PRINT "GO";"L";L;"
?": RETURN
60130  FOR D = A TO B + 1 STEP  - 1:X =  INT (W /
10):Y = W - 10 * X + 48: IF W = 0 THEN Y = 32
60140  POKE D,Y:W = X: NEXT D: IF W = 0 THEN  RETU
RN
60150  PRINT "INSERT";W%(J);"L";L: RETURN
60160  LET F = 2049:M = LN - IN
60170  LET A = F:M = M + IN
60180  LET F =  PEEK (A) +  PEEK (A + 1) * 256:L =
  PEEK (A + 2) +  PEEK (A + 3) * 256:A = A + 3:G =
 L < 6E4
60190  RETURN
60200  LET S = 0
60210  LET V = 0:A = A + 1:B = A:C =  PEEK (A): IF
 C = 0 THEN  GOSUB 60170: ON G + 2 GOTO 60210,6019
0
60220  IF C <  > 171 AND C <  > 176 AND C <  > 196
 AND C <  > S GOTO 60200
60230  LET A = A + 1:C =  PEEK (A) - 48: IF C =  -
 16 GOTO 60230
60240  IF C >  = 0 AND C < 9 THEN V = V * 10 + C:
GOTO 60230
60250  LET S = 44:A = A - 1: RETURN
60260  DEL 60000,60270
60270  END
```

Listing 1

```
]
```

APPLE II Applesoft Version of Jim Butterfield's Resequence program.

CLASSIFIED INDEX FOR ISSUES 7 TO 12

TIM, KIM, SYM, AND AIM

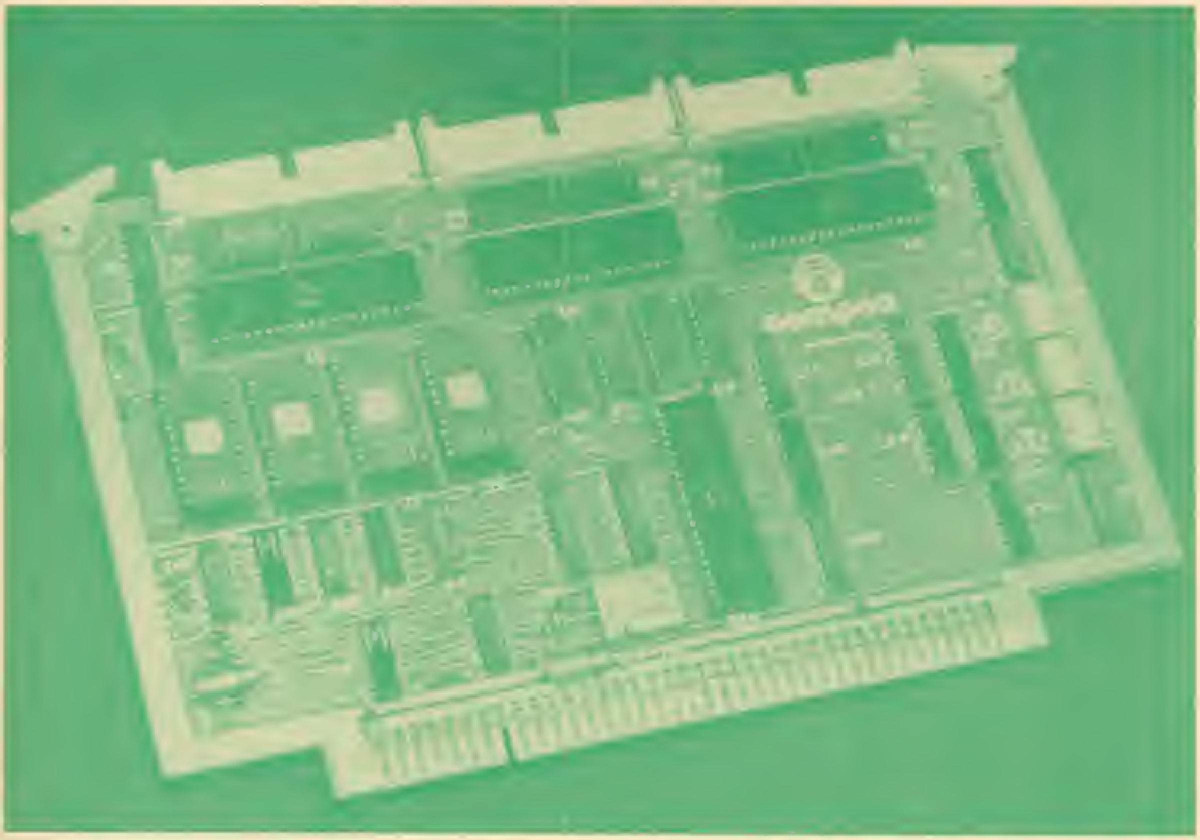
PET

compas
microsystems

224 S. E. 16th Street
P.O. Box 687
Ames, Iowa 50010
Phone 515-232-8187

single board computers

CSB I



THE 6500 BASED SINGLE BOARD COMPUTER YOU HAVE BEEN LOOKING FOR

An industrial quality 6500 based single board computer produced by the company known for quality 6500 hardware and software. CSB 1 includes a 6502 processor, four ROM/EPROM sockets, 2K bytes RAM(2114), two PIA(6520) and one VIA(6522) chips. All addresses are switch selectable and a single +5 volt power supply is needed. Board size is 6 by 9.75 inches. The gold plated edge connector adheres to the SYSTEM 65 bus standard. Single quantity price is $595 with substantial quantity discounts available. Other CSB board products are available or in design. Call us today for further information on our CSB family or any of our other 6500 support products.

Speakeasy Software™

FINANCIAL ANALYSIS
SPORTSTRIVIA

TRANSACTIONAL ANALYSIS
WARLORDS • KIDSTUFF
BULLS AND BEARS
MICROTRIVIA

Attention Dealers!
PET VERSIONS
Available internationally through
Byte
930 W. Maude, Sunnyvale,
California 94086

APPLE • PET • TRS~80

Now available at over 1,000 stores worldwide!

Speakeasy Software LTD.
Box 1220, Kemptville, Ontario, Canada, K0G 1J0
(613) 258-2451